T.C.
MUĞLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

KÜMELEME YAKLAŞIMI İLE MODEL TABANLI TEST ÖNCELİKLERİNİN BELİRLENMESİ

DOKTORA TEZİ

NİDA GÖKÇE

OCAK 2012
MUĞLA

N. GÖKÇE MATEMATİK ANABİLİM DALI DOKTORA TEZİ MUĞLA 2012

T.C.
MUĞLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

KÜMELEME YAKLAŞIMI İLE MODEL TABANLI TEST ÖNCELİKLERİNİN BELİRLENMESİ

DOKTORA TEZİ

NİDA GÖKÇE

OCAK 2012
MUĞLA

MUGLA ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

TEZ ONAYI

NİDA GÖKÇE tarafından hazırlanan **KÜMELEME YAKLAŞIMI İLE MODEL TABANLI TEST ÖNCELİKLERİNİN BELİRLENMESİ** tezinin, 27/12/2011 tarihinde aşağıdaki jüri tarafından Matematik Anabilim Dalı'nda doktora derecesi için gerekli şartları sağladığı oybirliği/~~oyçokluğu~~ ile kabul edilmiştir.

TEZ SINAV JURİSİ

Prof.Dr. Fevzi BELLİ* (Jüri Başkanı) İmza

Yazılım Mühendisliği Bölümü, Paderborn Üniversitesi, Almanya

Yrd. Doç. Dr. B. Taner DİNÇER** (Üye) İmza:

Bilgisayar Mühendisliği Bölümü, Muğla Üniversitesi, Muğla

Prof. Dr. Mehmet SEZER (Üye) İmza :

Matematik Anabilim Dalı, Muğla Üniversitesi, Muğla

Yrd. Doç. Dr. Bekir TANAY(Üye) İmza:

Matematik Anabilim Dalı, Muğla Üniversitesi, Muğla

Doç. Dr. Dursun AYDIN (Üye) İmza :

İstatistik Anabilim Dalı, Muğla Üniversitesi, Muğla

BÖLÜM BAŞKANLIĞI VE FEN BİLİMLERİ ENSTİTÜSÜ ONAYI

Prof. Dr. İlkay KUŞCU İmza

Fen Bilimleri Enstitüsü Müdürü, Muğla Üniversitesi, Muğla

Prof. Dr. Mehmet SEZER İmza

Matematik Anabilim Dalı, Muğla Üniversitesi, Muğla

Yrd. Doç. Dr. B. Taner DİNÇER İmza

Danışman, Bilgisayar Mühendisliği, Muğla Üniversitesi, Muğla

Savunma Tarihi: 27/12/2011

---

*Jüri Başkanının ismi birinci sırada

**Danışmanın ismi ikinci sırada

Tez çalışmalarım sırasında elde ettiğim ve sunduğum tüm sonuç, doküman, bilgi ve belgelerin tarafımdan bizzat ve bu tez çalışması kapsamında elde edildiğini; akademik ve bilimsel etik kurallarına uygun olduğunu beyan ederim. Ayrıca, akademik ve bilimsel etik kuralları gereği bu tez çalışması sırasında elde edilmemiş başkalarına ait tüm orijinal bilgi ve sonuçlara atıf yapıldığını da beyan ederim.

Nida GÖKÇE

Tarih:27/12/2011

ÖZET

KÜMELEME YAKLAŞIMI İLE MODEL TABANLI TEST ÖNCELİKLERİNİN BELİRLENMESİ

Nida GÖKÇE

Doktora Tezi, Matematik Anabilim Dalı

Danışman: Yrd. Doç. Dr. B.T. DİNÇER

Ocak 2012, 120 sayfa

Bu çalışmada test önceliklendirme probleminin çözümüne yönelik, yazılımın kaynak kodu ve test geçmişine ait hata sayısı gibi ön bilgiler bulunmadığı durumlarda test dizilerini hata ortaya çıkarma potansiyellerine göre sıralamayı amaçlayan kümelemeye dayalı yeni bir test önceliklendirme yöntemi geliştirilmiştir. Bu yöntem yazılımın sunduğu hizmetlerin kullanıcı tarafından kullanım modelini betimleyen olay dizisi grafiklerinden Çinli postacı problemi esasına göre üretilen test takımlarına uygulanmıştır. Olay olarak adlandırılan sistem bileşenlerini nitelendirebilmek için modeller ve test takımları üzerinden üç grupta farklı faktörler tanımlanmıştır. Tanımlanan faktörler yardımıyla birer vektör olarak temsil edilen olaylar klasik ve bulanık kümeleme algoritmaları ile kümelenmiştir. Oluşturulan kümelere önem dereceleri atanmış ve önemlilik değeri yüksek olaylardan oluşan kümedeki olayların bulunduğu test dizilerinin önce test edilmesinin hata yakalama açısından önemli olduğu gösterilmiştir.

Önerilen önceliklendirme yöntemi bir Web yazılımının testi sürecine uygulanmıştır. Farklı hiyerarşik düzeylerde birden çok ODG graf ile modellenen sistem üzerinden önce her graf için ayrı ayrı, daha sonrada detaylandırılmış model (tüm grafları içeren) üzerinden tek bir test takımı üretilerek önceliklendirilmiştir. Önceliklendirilmiş test takımı üzerinden gerçekleştirilen test sonucunda modüldeki hatalar ortaya çıkarılmıştır. Test sonucunda ortaya çıkarılan hata sayıları kullanılarak önerilen önceliklendirme metotlarının hata yakalama başarımları değerlendirilmiştir. Kümeleme ile test önceliklendirme metotlarının performanslarını değerlendirmek için, hata yakalayan olay kapsama, tekrarlı hata yakalama, tekrarsız hata yakalama, tüm hataları ortaya çıkarma hızı gibi dört yeni değerlendirme kriteri kullanılmıştır. Ticari bir uygulamanın testi üzerinden yapılan denemeler sonucunda kümeleme ile test önceliklendirme yöntemlerinin sistemdeki hataları erken ortaya çıkarmada başarılı olduğu görülmüştür.

Anahtar Kelimeler: Kara Kutu Testi, Test Önceliklendirme, Kümeleme**,** Uyarlamalı Yarışmacı Öğrenme, Bulanık C-Ortalamalar.

ABSTRACT

DETERMINATION OF MODEL BASED TEST PRIORITIES BY CLUSTERING APPROACH

Nida GÖKÇE

Doctoral Thesis (PhD), Department of Mathematics

Supervisor: Asst. Prof. Dr. B.T. DİNÇER

January 2012, 120 pages

In this study, a new test prioritization method based on clustering was developed for solving the problem of test case prioritization. This method is aimed at sorting test sequences according to their potentials of revealing faults in the cases software source code and prior information for fault are unavailable. The method was applied to the suites of tests generated on the basis of Chinese Postman Problem which is one of the event sequences graphs describing usage model for services of the software by user. In order to characterize the components of the system called events, the factors in three different groups were defined by using ESG models and test sequences. Represented as vectors with the help of the defined factors, the events were clustered by hard and fuzzy clustering algorithms. The importance degrees were assigned to the resulting clusters. And it was shown to be important in terms of detecting failures to test earlier the test sequences where there are events of the cluster composed of events of high importance.

Suggested test prioritization method was applied to the test process for a web based application. The software system was modeled using more than one ESG at different hierarchical levels. Firstly, test suites were generated for each graph separately, then only one test suite was generated by the model detailed in such a way to cover all models and the method was applied to these test suites. As a result of the test conducted faults in the system were revealed. Four new performance evaluation criteria such as coverage event capturing failure, capturing repetitive failure, capturing non-repetitive failure and rate of revealing all failures, which expose fault detection success of test sequences, are used for the evaluation of the proposed clustering-based test prioritization method. The results of the experimental analyses performed using a commercial application has shown that the fault detection performance of test prioritization methods can be increased considerably by using the approach.

Key Words: Black Box Testing, Test Prioritization, Clustering, Adaptive Competitive Learning, Fuzzy c-Means.

ÖNSÖZ / TEŞEKKÜR

Yazılım testi sürecinde, zaman ve maliyet kısıtı altında bir yazılımın tamamını test etmenin mümkün olmadığı durumlarda, yazılıma ait olası test durumlarını sistemdeki hataları erken ortaya çıkaracak şekilde önceliklerine göre sıralamak gerekir. Bu problem literatürde test önceliklendirme problemi olarak bilinir. Mevcut test önceliklendirme yöntemlerinin büyük bir çoğunluğu önceliklendirme için test edilen sistemin kaynak koduna veya daha önceki çalıştırılmalarına ait hata bilgilerine ihtiyaç duyarlar. Bu tez çalışmasında test edilecek sisteme ait ön bilgi bulunmadığı durumlarda derlenmiş kod üzerinden kullanıcı harekelerini baz alan, model tabanlı ve kümelemeye dayanan test önceliklendirme metotları geliştirilmiştir. Geliştirilen bu metotlar web tabanlı bir sistemin test sürecine uygulanmıştır.

Tez konumu belirleme ve yazılım testi konusunda çalışmaya karar vermemde büyük etkisi olan Paderborn Üniversitesi, Elektrik Mühendisliği ve Bilgi Teknolojileri Enstitüsü, Yazılım Mühendisliği Kürsü Başkanı Prof. Dr. Fevzi BELLİ'ye doktora sürecimde ki yardımlarından ve tüm desteğinden dolayı sonsuz saygı ve şükranlarımı sunuyorum. Doktora çalışmamın bir parçası olarak Almanya'da bulunduğum tüm süreler boyunca bilgi ve becerilerini benimle paylaştıkları için çalışma arkadaşlarım Axel HOLLMANN, Michael LINSCHULTE, Mutlu BEYAZIT ve Sascha PADBERG'e ve kendimi sıcak bir aile ortamında hissetmemde büyük katkıları olan değerli Bettina BELLİ'ye sonsuz teşekkürlerimi sunuyorum.

Bilgi ve önerileriyle tez çalışmamın şekillenmesine yardımcı olan Prof. Dr. Mübariz EMİNOV'a emeği ve katkıları için çok teşekkür ediyorum. Danışmanım Yrd. Doç. Dr. Bekir Taner DİNÇER'e eleştiri ve yönlendirmeleriyle tezime olan katkılarından dolayı sonsuz teşekkürlerimi sunuyorum. Değerli hocam Öğr. Gör. Mehmet KARAHASAN'a tüm yardımlarından dolayı çok teşekkür ediyorum. Tez izleme komitemde yer alan ve değerli eleştirileriyle tezime yön veren Prof. Dr. Mehmet SEZER ve Yrd. Doç. Dr. Bekir TANAY'a çok teşekkür ediyorum.

Tez çalışmam boyunca gerek bilgi ve becerileriyle gerekse manevi destekleriyle yanımda olan başta Araş. Gör. Aytaç PEKMEZCİ ve Araş. Gör. Nevin GÜLER DİNÇER olmak üzere tüm arkadaşlarıma çok teşekkür ediyorum.

Ve son olarak tüm yaşamım boyunca yanımda olan, maddi manevi desteklerini bir an bile esirgemeyen başta babam Vahit GÖKÇE ve annem Ayşe GÖKÇE olmak üzere tüm aileme varlıkları için sonsuz şükranlarımı sunuyorum.

Sevgili Aileme

İÇİNDEKİLER

ÖZET ..... **iv**
ABSTRACT ..... **v**
ÖNSÖZ / TEŞEKKÜR ..... **vi**
İÇİNDEKİLER ..... **viii**
ŞEKİLLER DİZİNİ ..... **x**
TABLOLAR DİZİNİ ..... **xi**
EKLER DİZİNİ ..... **xii**
SEMBOLLER VE KISALTMALAR DİZİNİ ..... **xiii**
1. GİRİŞ **1**
1.1. Amaç ve Kapsam 1
1.2. Kaynak Özetleri 4
1.2.1. Yazılım testi 4
1.2.2. Yazılım test teknikleri 6
1.2.3. Test önceliklendirme 7
1.2.4. Sistem modelleme 11
1.2.5. Test takımı üretme 12
1.2.6. Kümeleme analizi 13
1.2.7. Yumuşak hesaplamaya dayalı kümeleme 14
1.2.7.1. Klasik kümeleme – yapay sinir ağları 15
1.2.7.2. Bulanık kümeleme – bulanık c-ortalamalar 17
1.2.8. Kümeleme tabanlı test önceliklendirme 17
1.3. Tezin Katkısı 18
1.4. Tezin Organizasyonu 20
2. MALZEME VE YÖNTEM **21**
2.1. Modelleme - Olay Dizisi Grafikleri 22
2.2. Test Takımı Üretme 26
2.3. Olayları Nitelendiren Faktörler 27
2.3.1. Tek model üzerinden tanımlanan faktörler 28
2.3.2. Test dizileri üzerinden tanımlanan faktörler 30
2.3.3. Farklı düzeylerdeki modeller için tanımlanan faktörler 32

2.4. Kümeleme Tabanlı Test Önceliklendirme 33
2.4.1. Gözlem ortalamalarına dayalı test önceliklendirme 35
2.4.2. Klasik kümeleme – yapay sinir ağlar 38
2.4.3. Yarışmacı öğrenme 40
2.4.3.1. Uzaklığa dayalı yarışmacı öğrenme 41
2.4.3.2. Açıya dayalı yarışmacı öğrenme 41
2.4.4. Yapay sinir ağ tabanlı test önceliklendirme 44
2.4.5. Bulanık kümeleme – bulanık c-ortalamalar 46
2.4.6. Bulanık c-ortalamalara dayalı test önceliklendirme 47
2.5. Sonuçlar 49
2.5.1. Maliyetlerin karşılaştırılması 50
3. BULGULAR VE İRDELEME 53
3.1. Uygulama I 54
3.1.1. Test Maliyeti 58
3.1.2. Kümeleme tabanlı test önceliklendirme 59
3.1.3. Farklı düzeylerde test önceliklendirme stratejileri 61
3.1.4. Sonuçlar 62
3.2. Uygulama II 63
3.2.1. Farklı düzeylerde modelleme 63
3.2.2. Test önceliklendirme 64
3.2.3. Test önceliklendirme yöntemlerinin başarımlarının karşılaştırılması 66
3.2.4. Sonuçlar 67
3.2.4.1. Hata yakalayan olay kapsama (HYOK) kriteri 71
3.2.4.2. Tekrarlı hata yakalama (TRHY) kriteri 74
3.2.4.3. Tekrarsız hata yakalama (TZHY) kriteri 75
3.2.4.4. Tüm hataları ortaya çıkarma hızı (HOÇH) 77
4. TARTIŞMA 79
5. SONUÇLAR VE ÖNERİLER 81
EKLER 91
ÖZGEÇMİŞ 119

ŞEKİLLER DİZİNİ

Şekil 1.1. Yazılım geliştirme sürecinde V-modeli (Armour, 2003) 5
Şekil 1.2. Hataların ortaya çıkma süreci (Belli, 2008) 6
Şekil 1.3. Mevcut test süreci önceliklendirme tekniklerine genel bakış 10
Şekil 1.4. Önerilen yöntemin farkı ve avantajları 19
Şekil 2.1. Kümeleme ile model tabanlı test önceliklendirme yöntemi uygulama adımları 22
Şekil 2.2. Basit bir ODG (Budnik, 2006) 23
Şekil 2.3. ODG_1, Birinci düzey graf; ODG_2, İkinci düzey graf; ODG_3, Detaylandırılmış graf (Budnik, 2006) 25
Şekil 2.4 (a) Çim biçme makinesi ve kontrol paneli (b) Kesme biriminin beklenen hareketlerini temsil eden model. (c) İstenmeyen hareketleri temsil eden sistem modeli (Budnik, 2006) 25
Şekil 2.5. Yarışmacı öğrenme ağ yapısı (Fu,1994) 39
Şekil 3.1. ISELTA'nın özel indirimler / fırsatlar modülünün ekran görüntüsü 53
Şekil 3.2. IS: Özel ındirim modülü birinci düzeydeki temel ODG modeli 56
Şekil 3.3. *ICD:* Eksik veri girişini temsil eden ODG modeli 56
Şekil 3.4. *ECD:* Tam veri girişini temsil eden ODG modeli 57
Şekil 3.5. CD: Veri değiştirme olaylarını temsil eden ODG modeli 57
Şekil 3.6. *SD:* Tarih seçme adımlarını temsil eden ODG modeli 57
Şekil 3.7. Test önceliklendirme metotlarının HYOK kriterine göre ideal sıralama ile karşılaştırmalı grafiği 73
Şekil 3.8. Test önceliklendirme metotlarının HYOK kriterine göre ideal sıralamadan sapma grafiği 73
Şekil 3.9. Test önceliklendirme metotlarının TRHY kriterine göre ideal sıralama ile karşılaştırmalı grafiği 74
Şekil 3.10. Test önceliklendirme metotlarının TRHY kriterine göre ideal sıralamadan sapmalarının grafiği 75
Şekil 3.11. Test önceliklendirme metotlarının TZHY kapasitesine göre ideal sıralama ile karşılaştırmalı grafiği 76
Şekil 3.12. Test önceliklendirme metotlarının TZHY kapasitesine göre idealden sapma grafiği 77

TABLOLAR DİZİNİ

Tablo 1.1. Test önceliklendirme probleminin tanımı 9
Tablo 2.1. Faktör listesi 27
Tablo 2.2. Ham veri matrisi 34
Tablo 2.3. Beşinci faktörün dönüşümü sonrası elde edilen veri seti 35
Tablo 2.4. Standartlaştırılmış veri seti 36
Tablo 2.5. Gözlem ortalamalarına göre olay sıraları ve önemlilik indeksleri 37
Tablo 2.6. Gözlem ortalamalarına dayalı test önceliklendirme sonuçları 38
Tablo 2.7. Uyarlamalı yarışmacı öğrenme (UYO) algoritması 43
Tablo 2.8. Yarışmacı öğrenme ile kümeleme sonuçları 44
Tablo 2.9. Yarışmacı öğrenmeye dayalı test önceliklendirme sonuçları 45
Tablo 2.10. Bulanık c-ortalamalar ile kümeleme sonuçları 47
Tablo 2.11. Bulanık c-ortalamalara dayalı test önceliklendirme sonuçları 49
Tablo 2.12. Karşılaştırmalı sonuçlar 49
Tablo 2.13. Friedman testi sonuçları 50
Tablo 2.14. Kendall Tau testi sonuçları 50
Tablo 2.15. Maliyet açısından karşılaştırmalı sonuçlar 52
Tablo 3.1. Modelde yer alan olayların açıklamaları 57
Tablo 3.2. Modellerden üretilen test dizilerinin listesi 58
Tablo 3.3. Her düzey için belirlenen test maliyetleri 59
Tablo 3.4. Test dizilerinin kümeleme ile önceliklendirme sonuçları 60
Tablo 3.5. Düzeylere göre karşılaşılan hata sayısı 62
Tablo 3.6. Düzeylere göre karşılaşan hataların listesi 62
Tablo 3.7. Test dizilerinin önceliklendirme ve sıralama sonuçları 65
Tablo 3.8. Test sonucunda ortaya çıkarılan hatalar 69
Tablo 3.9. HYOK kriterine göre ideal durum ile önceliklendirme metotları arasındaki ilişki analizi sonuçları 72
Tablo 3.10. TRHY kriterine göre ideal durum ile metotlar arasındaki ilişki analizi sonuçları 75
Tablo 3.11. TZHY kriterine göre ideal durum ile önceliklendirme metotları arasındaki ilişki analizi sonuçları 76
Tablo 3.12. Sistemde karşılaşılan tüm hataların ortaya çıkarılma maliyeti 78
Tablo 4.1. Önceliklendirme metotlarının genel başarım tablosu 79

EKLER DİZİNİ

Ek 2. 1. Örnek ODG'ler için Test Önceliklendirme Uygulamaları 91
Ek 3. 1. Uygulama II'de Kullanılan Modeller 98
Ek 3. 2. Uygulama II'de Kullanılan Test Dizileri 100
Ek 3. 3. Yazılıma Eklenen Hatalar 111
Ek 3. 4. Kendall Tau-b İstatistiklerinin Hesaplanmasında Kullanılan Veri Tabloları 113

SEMBOLLER VE KISALTMALAR DİZİNİ

f	PT'den Reel sayılara bir fonksiyon
V	Olaylar kümesi
E	Ark kümesi
a,b	Olaylar
Σ	Olay çiftleri kümesi
ab, ba	Ark, geçiş
[	Başlangıç/giriş olayı
]	Bitiş/final olayı
$Avrf(x_i)$	i. olayın ortalama kullanılma sıklığı
n	Gözlem, olay sayısı
l	Uzunluk
x_i	i. olay
$l(TOD_q)$	q. test dizisinin uzunluğunu
$f_q(x_i)$	q. test dizisinde x_i olayının kullanılma sıklığı
d	i. olayın kullanıldığı test sayısı
$\in$	Eleman
N	Doğal sayılar
$Q, r, n, i, j \in N$	Q, r, n, i, j Doğal sayılar kümesinin birer elemanı
μ	Örneklem ortalaması
σ^2	Örneklem varyansı
$Imp(x_i)$	i. olayın önemlilik indeksi
$ImpS(x_i)$	i. olayın önemlilik sırası
e	Toplam olay sayısı
$PrefD(TOD_q)$	q. test dizisinin öncelik değeri
p	Faktör sayısı
c	Optimal küme sayısı
$\mathbb{R}^p$	p boyutlu Reel vektör uzayı
X	Veri seti
w_k	Kazanan ağırlık vektörü

Δw_k	Ağırlık vektörü güncelleme değeri
η	Öğrenme oranı
$\widetilde{w}_w$	Birim uzunluktaki kazanan ağırlık vektörü
$\tilde{x}_i$	Birim uzunluğa dönüştürülmüş i. olay
D_k	k. kümenin sınıflandırma hatası
S_k	k. küme
V_{sv}	Optimal küme sayısı üretme algoritması
$k(0)$	Başlangıç küme sayısı
L	Optimal küme sayısı
t	Döngü sayısı
T_{max}	Maksimum döngü sayısı
$l\,(\bar{x}_k)$	Ortalama vektör uzunluğu
$S_k^{(YÖ)}$	Yarışmacı öğrenme ile elde edilen kümeler
U	Üyelik değerleri matrisi
V	Bulanık küme merkezlerini
$d(v_k, x_i)$	k. küme merkezi ile i. olay arasındaki öklit uzaklığı
$S_k^{(BCO)}$	Bulanık c-ortalamalar algoritması sonucu elde edilen kümeler
$\mu_{S_k}(x_i)$	i. olayın k. kümeye aitlik derecesini veren üyelik değeri
α	Her bir işlem adımını gösteren tıklama maliyeti
β	Bir test dizisinin test için başlangıç maliyeti
l_{cov}	Ortalama test dizisi uzunluğu
$\#\,TOD$	Test dizisi sayısı
NIST	**N**ational **I**nstitute of **S**tandarts and **T**echnology
ODG	Olay Dizisi Grafikleri
ESG	Event Sequence Graphs
ÇPP	Çinli Postacı Problemi
ISELTA	Isık's System for Enterprise-Level Web-Centric Tourist Applications
TÖP	Test Önceliklendirme Problemi
UML	Unified Modelling Language / Birleşik Modelleme Dili
TT	Test Takımı
PT	TT'nin permütasyonlarının kümesi
YÖ	Yarışmacı Öğrenme

CL	Competitive Learning
BCO	Bulanık C-Ortalamalar
YSA	Yapay Sinir Ağları
ADALINE	ADAptive LINear Element
FCM	Fuzzy C-Means
OÇ	Olay Çifti
EP	Event Pair
HOÇ	Hatalı Olay Çifti
FEP	Faulty Event Pair
OD	Olay Dizisi
ES	Event Sequence
TOD	Tam Olay Dizisi
CES	Complete Event Sequence
DD	Denge Derecesi
UYO	Uyarlamalı Yarışmacı Öğrenme
BCO	Bulanık C-Ortalamalar
FCM	Fuzzy C-Means
GO	Gözlem Ortalamaları
ISELTA	Isık's System for Enterprise-Level Web-Centric Tourist Applications
ICD	Enter InComplete Data
ECD	Enter Complete Data
CD	Change Data
SD	Select Date
IS	ISELTA Special module
HYOK	Hata Yakalayan Olay Kapsama Kriteri
TRHY	TekRarlı Hata Yakalama Kriteri
TZHY	TekrarsıZ Hata Yakalama Kriteri
HOÇH	Tüm Hataları Ortaya Çıkarma Hızı

1. GİRİŞ

Yazılım testi, yazılım sektöründe önemli kalite kontrol tekniklerinden biridir (Beizer, 1990; Binder, 2000; Mathur, 2008). Yazılımların kullanıcıya sunulmadan önce doğru çalışıp çalışmadıklarının test edilmesi şayet varsa yazılım hatalarından arındırılması gerekir. Yazılımın kalitesi, kullanıcı gereksinimlerinin doğru ve eksiksiz gerçekleştirilmesiyle doğru orantılıdır. Bu nedenle yazılım testinin yazılım geliştirme sürecinin bir parçası olarak ele alınması zorunluluktur. Yazılım testi, yazılım geliştirme sürecinde çok zaman harcanması gereken ve oldukça pahalı bir süreçtir (Beizer, 1990; Craig ve Jaskiel, 2002). 2002'de Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından yapılan bir araştırma yazılım hatalarının Amerikan ekonomisine yıllık maliyetinin yaklaşık 59.5 milyar dolar olduğunu göstermiştir (Anonim, 2002). Zaman ve maliyet kısıtı olduğu durumlarda yazılımın tamamını test etmek mümkün olmayabilir. Bu durumda yazılımda hata ortaya çıkması muhtemel kısımların önce test edilmesi gerekir. Test durumlarını hataları erken yakalayabilme potansiyeline göre sıralama problemi literatürde test durumlarını önceliklendirme (*ing. test case prioritization*) problemi olarak adlandırılmaktadır. Mevcut test önceliklendirme yöntemleri regresyon testlerine yöneliktir ve sistemin daha önceki kullanımlarına ilişkin bilglere ihtiyaç duyarlar. Bu çalışmada test önceliklendirme problemleminin çözümüne yönelik, mevcut yaklaşımlardan farklı olarak yazılımın kaynak kodu ve ön bilgi olarak hata bulma bilgisi olmadığı durumlarda test dizilerini sıralamaya imkan sağlayan kümelemeye dayalı yeni bir test önceliklendirme yöntemi önerilmiştir.

1.1. Amaç ve Kapsam

Yazılım testinde öncelikli amaç, kodda veya sistem işleyişinde var olan hataları ortaya çıkarmak ve sistemi mümkün olduğunca hızlı bu hatalardan arındırmaktır. Yazılım geliştirme sürecinde yazılımın hangi kısımlarının önce test edileceği, test

işlemine nereden başlanacağı, özellikle zaman ve diğer kaynakların kısıtlı olduğu durumlarda kritik bir öneme sahiptir. Yazılım sistemlerini test etmek için çok sayıda test durumunun (veya test dizisinin) incelenmesi gerekebilir ve bunun bir maliyeti vardır. Bir "*test durumu* (*ing. test case*)*"* en basit anlamıyla bir yazılımın gerçekleştirmesi gereken bir fonksiyonunun işlem adımlarının ardışık bir dizisi olarak tanımlanabilir. Veya bir test durumu bir yazılımın birden fazla fonksiyonunu içine alabilir yada tüm yazılımı da kapsayabilir. Bir "test süreci" ise bu test durumlarının elle veya otomatik olarak kontrol edilmesi sürecidir. Test dizilerinin oluşturulmasında / üretilmesinde temelde iki yaklaşım mevcuttur: Beyaz kutu (*ing. white box*), yani sistemin kaynak kodu üzerinden geliştirilen test yöntemleri; kara kutu (*ing. black box*), yani kaynak koda ulaşmanın mümkün olmadığı durumlarda derlenmiş kod üzerinden geliştirilen test yöntemleri. Bu çalışmada derlenmiş kod üzerinden çalıştırılan bir yazılımın kullanıcı hareketlerini gösteren bir modeli üzerinden test süreci ele alınmaktadır. Derlenmiş kod üzerinden gerçekleştirilen testler sistemin fonksiyonel davranışları veya sistem yapısının modellenmesine göre sırasıyla yönergeye dayalı (*ing. specification-oriented*) ve uygulamaya dayalı (*ing. implementation-oriented*) test teknikleri olarak iki grupta incelenebilir. Özetle bu çalışmadaki test tekniği kara kutu test teknikleri grubunda uygulamaya dayalı bir yöntemdir.

Yazılımlar sağladıkları fonksiyonlarla sınırlı kullanım imkanı verirler. Kullanıcılar her bir fonksiyonu ardışık halde kullanarak yazılımın sunduğu hizmetlerden faydalanabilirler. Bu nedenle, yazılım piyasaya sürülmeden önce olası tüm fonksiyon dizisi kombinasyonları, yani hizmetleri, test edilmek zorundadır. Test işlemi yazılımın kaynak kodu üzerinden veya sistemin kullanıcı hareketlerini ifade eden bir model üzerinden yürütülebilir. Yazılım firmalarının gizlilik gereği yazılımın kaynak kodunu açıklamadığı durumlarda hizmetleri betimleyen bir model üzerinden yazılımı test etmek mümkündür. Literatürde bu tür teknikler model tabanlı test yöntemleri olarak adlandırılmaktadır (*ing. model based testing*). Burada modelden kasıt, bir kişinin yazılımı kullanım modelidir. Bu çalışmanın genelinde "olay (*ing. node, event*)" olarak adlandırılacak bileşenlerin birbirleri ile ilişkilerini modellemek için literatürdeki sistem modelleme yaklaşımları arasından olay dizisi grafikleri (ODG) (*ing. event sequence graphs, ESG*) kullanılmıştır (Belli, 2001; Belli vd., 2006). ODG'ler sistem bileşenlerinin olay, kullanıcı hareketlerinin ark (*ing. vertex, arc*)

olarak gösterildiği yönlü graflardır. Bu model üzerinden üretilen sonlu sayıdaki kesikli olaylar dizisi test durumu olarak adlandırılmaktadır. Kesikli olaylar dizisi, girdi olarak kullanıcı aktivitelerini kullanan, çıktı olarak da beklenen sistem cevabını üreten ardışık bir dizi olarak tanımlanabilir. Yazılım testleri, yazılımın istenilen davranışları baz alınarak gerçekleştirilirse pozitif test (*ing. positive testing*), istenmeyen davranışlar baz alınarak gerçekleştirilirse negatif test (*ing: negative testing*) olarak adlandırılır. Yazılım geliştirme sürecinde programcının kod yazarken yaptığı "yanlışlık" (*ing. error*) sonucu kod içinde "hata" (*ing. fault*) oluşur. Bu hatanın bir sonucu olarak yazılımın gözlenen yanlış davranışı "başarısızlık" (*ing. failure*) olarak adlandırılır. Bu çalışma pozitif testler üzerinden yürütülmüş ve çalışma boyunca yazılımın istenmeyen davranışlar sergilemesi veya istenen davranışları yerine getirememesi durumları "*hata"* olarak adlandırılmıştır.

Model üzerinden üretilen tüm test durumlarının kümesi ise test takımı (*ing. test suite*) olarak adlandırılmaktadır. Böylesi bir sistem sonsuz sayıda test durumu (*ing. test cases*) / test dizisi (*ing. test sequence*) üretebilir. Ancak maliyet kısıtı nedeniyle test takımını oluşturmak için test durumlarının optimal bir şekilde seçilmesi/üretilmesi gerekir. Bu çalışmada test durumlarını üretmek için Çinli Postacı Problemi (ÇPP) (*ing. Chinese Postman Problem*) esas alınmıştır. ÇPP bir graftaki her arktan en az bir kez geçerek en kısa yolu bulma problemidir. ÇPP ile amaç yazılımın istenilen davranışlarını modelleyen bir ODG üzerindeki tüm arkları ziyaret etmek koşuluyla en kısa yolu içeren test takımını üretmektir.

Test takımını oluşturan test durumlarının sayısı yazılımın kapsamıyla doğru orantılı olarak artmaktadır. Test sürecinde bir test durumunu oluşturan olaylar dizisi tek tek çalıştırılarak hata ortaya çıkarılmaktadır. Bir test dizisinin çalıştırılması sırasında bir hata ortaya çıkarsa bu hata düzeltilip test dizisi tekrar işletilmelidir. Ancak her test dizisinin başlatmanın ve dizideki her olayı işletmenin bir maliyeti vardır. Çoğu zaman bir yazılımın olası tüm fonksiyonlarını incelemek bütçe kısıtı altında mümkün olmayabilir. Bu durumda test bütçesini aşmayacak şekilde mümkün olduğunca çok test dizisini incelemek gerekir. Yazılım kalitesi açısından öncelikle incelenecek test dizilerinin ya yazılımın en sık kullanılan kısımları yada hataya en açık kısımları olması önemlidir. Yazılımın varsa hayati önemdeki parçaları yada hata meydana gelme olasılığı en yüksek olan kısımlarının önce test edilmesi yazılımın kalitesini ve

güvenilirliğini arttıracaktır. Dolayısıyla bu noktada karşılaşılan problemlerden biri test dizilerini sistemdeki hataları önce yakalayacak şekilde sıralayabilmektir. Bu problem literatürde test süreci önceliklendirme problemi (*ing. test case prioritization*) olarak bilinmektedir.

Literatürdeki model tabanlı test süreci önceliklendirme yöntemleri genellikle test dizilerini, kapsam kriterine (*ing. coverage criteria*) veya test dizisinin önceki testlerden elde edilen hata bilgilerine (*ing. regression testing*) göre sıralamakta ve her sıralama kriterini tek tek ele almaktadırlar (Goodenough ve Gerhart, 1975; Binder, 2000; Mathur, 2008). Ancak bu yöntemler sistemin önceki çalıştırmalarına ait hata bilgileri olmadığında ve eşit uzunluktaki test dizileri söz konusu olduğunda sıralama yapmakta yetersiz kalmaktadırlar.

1.2. Kaynak Özetleri

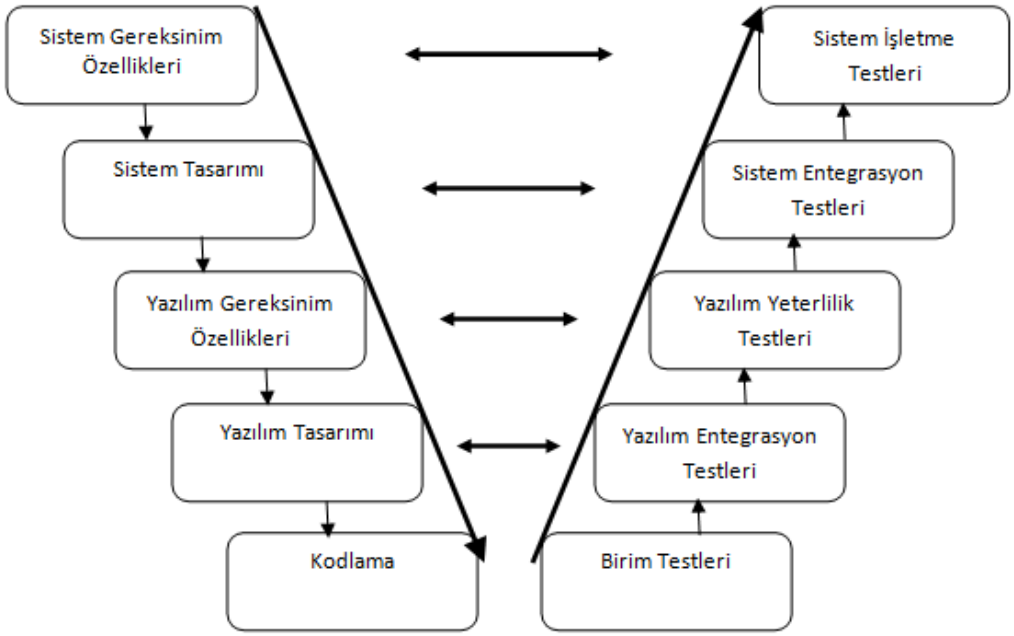
Bu bölümde çalışmada kullanılan yazılım testiyle ilgili genel kavramlar verilmiş ve test önceliklendirme problemi (TÖP) tanımlanmıştır. TÖP'nin çözümüne yönelik literatürdeki mevcut yöntemler verilmiştir. Ayrıca kümeleme analizi ve yumuşak hesaplama yöntemlerinden yapay sinir ağları ve bulanık kümeleme yaklaşımlarına ait literatür taraması verilmiştir.

1.2.1. Yazılım testi

Yazılım testi, yazılım geliştirme sürecinde çok zaman harcanması gereken oldukça pahalı bir süreçtir (Beizer, 1990; Craig ve Jaskiel, 2002). 2002 yılında NIST (**N**ational **I**nstitute of **S**tandarts and **T**echnology) tarafından yapılan bir araştırma yazılım hatalarının Amerikan ekonomisine yıllık maliyetinin yaklaşık 59.5 milyar dolar olduğunu göstermiştir. 2002'den günümüze bilişim sektöründeki gelişmeler dikkate alındığında yazılım testlerinin önemi ortaya çıkmaktadır. Yazılım testinde amaç yazılımların piyasaya sürülmeden önce mümkün olduğunca hatalardan arındırılmasıdır.

Yazılım testi yazılım geliştirme sürecinde gereksinimlerin belirlenmesi aşamasından yazılımın kullanıcıya sunulması aşamasına kadar her adımda önemli bir rol oynar. Ayrıca yazılımın kullanıcıya sunlumasını ardından bakım ve güncellenme sürecininde önemli bir parçasıdır. Yazılım geliştirme süreci boyunca gerçekleştirilen test tekniklerinin en yaygın gösterimi V-modeli olarak bilinen modeldir. Şekil 1.1'de verilen bu modelde sürecin her aşamasına karşılık bir test mevcuttur. "V" görünümlü bu modelde sol taraf üretim sürecine, sağ taraf ise test sürecine karşılık gelecek şekilde bir yol izlenir (Armour, 2003).

Bu çalışmada ele alınan test yöntemi yazılımın piyasaya sürülmeden önceki son test aşaması olan sistem işletme testleri grubunda ele alınabileceği gibi yazılımın kullanıma sunulduktan sonraki bakım sürecini de kapsamaktadır. Sistem testlerinin amacı sistemi mümkün olduğunca hatalardan arındırmak ve yazılımın kalite ve güvenilirliğini artırmaktır.



Şekil 1.1. Yazılım geliştirme sürecinde V-modeli (Armour, 2003)

Yazılım geliştirme sürecinde sistem gereksinimlerinin tanımlanmasından, kodlama sürecine kadar her süreçte hata ortaya çıkması mümkündür. Bu hatalar düzeltilmediği sürece bir sonraki süreçte başka hatalara neden olacak ve son aşamaya gelindiğinde bu hataların düzeltilme maliyeti artacaktır. Bu nedenle birim testleri, bütünleşme (entegrasyon) testleri, yeterlilik testleri ve sistem entegrasyon testlerinin

gerçekleştirilmesi önemlidir. Yazılım geliştirme sürecinin bir parçası olarak hataların gelişim süreci Şekil 1.2'de verilmiştir (Belli, 2008).

Şekil 1.2'de ilk sütun yazılımın doğru ve eksiksiz çalışması durumunu göstermektedir. Diğer durumlar hataları temsil etmektedir. Sürecin gelişimi sırasında hataların kümülatif artışı satır boyunca gözükmektedir. Kısaca son test hem yazılımın kalite ve güvenilirliği açısından hem de yazılım şirketlerinin prestijleri açısından oldukça önemlidir. Bu nedenle yazılımlar kullanıcıya sunulmadan önce test edilmek zorundadır.

Şekil 1.2. Hataların ortaya çıkma süreci (Belli, 2008)

Şekil 1.2'de ilk sütun yazılımın doğru ve eksiksiz çalışması durumunu göstermektedir. Diğer durumlar hataları temsil etmektedir. Sürecin gelişimi sırasında hataların kümülatif artışı satır boyunca gözükmektedir. Kısaca son test hem yazılımın kalite ve güvenilirliği açısından hem de yazılım şirketlerinin prestijleri açısından oldukça önemlidir. Bu nedenle yazılımlar kullanıcıya sunulmadan önce test edilmek zorundadır.

1.2.2. Yazılım test teknikleri

Yazılım testleri yazılım geliştirme sürecinde kullanıldıkları aşamalara göre farklılık göstermektedir (Beizer, 1990). Bu çalışmada ele alınan test teknikleri literatürde

kullanıcı kabul testleri (*ing. user acceptance testing*) olarak da adlandırılan yazılımın piyasaya sürülmeden önce son kontrollerinin gerçekleştirildiği sistem işletme testleri grubundandır. Yazılım test teknikleri temelde ikiye ayrılmaktadır. Kaynak kod üzerinden gerçekleştirilen beyaz kutu testleri ve kaynak koda ulaşılamadığı durumlarda derlenmiş kod üzerinden gerçekleştirilen kara kutu testleri (Myers, 1979; Malaiya, 1995; Beizer 1990; Qu vd., 2007). Beyaz kutu testleri ile yazılım kodundaki deyimler, akış denetimleri, koşullar v.b. elemanlar kontrol edilerek hem kaynak kod hem de derlenmiş kod test edilir. Bu testler, yazılımı geliştiren ekip tarafından gerçekleştirilirken, kara kutu testleri yazılım ekibi olabileceği gibi ekip dışı kişiler tarafından da sistem modeline dayalı veya sistemin dış arayüzlerini esas alarak gerçekleştirilebilir. Kara kutu testlerinde, yazılımın programatik yapısı, tasarımı veya kodlama tekniği hakkında herhangi bir ön bilgiye ihtiyaç duyulmaz. Kara kutu testlerinde yazılımın geliştirilme amacına uygun olup olmadığı, kullanıcı ihtiyaçlarına cevap verip vermediği ve işlevselliğini gerçekleştirip gerçekleştirmediği test edilir. Kısacası bu testler sistemin tümüne yönelik işlevlerin doğru yürütülüp yürütülmediğini test etmek için kullanılırlar ve koda ihtiyaç duymadıkları için beyaz kutu testlerine göre daha az maliyet gerektirirler (Malaiya, 1995; Myers,1979).

Derlenmiş kodlar üzerinden gerçekleştirilen kara kutu testleri yönergeye ve uygulamaya dayalı testler olarak ikiye ayrılır. Yönergeye dayalı test yöntemleri sistemin önceden tanımlanan yönergelere uygunluğunu test ederken uygulamaya dayalı test yöntemleri ise yazılımın kullanıcı modeli üzerinden test işlemini gerçekleştirir (Chen ve Subramaniam, 2002). Yazılımın kullanıcı modeli sonlu durum makineleri (*ing. Finite State Machines*), durum kartları (*ing. State Charts*), UML (*ing. **U**nified **M**odelling **L**anguage*) diagramları, veya olay dizisi grafları (ODG) yardımıyla oluşturulabilir (Chow, 1978; Yannakakis ve Lee, 1995; Acharya vd., 2010).

1.2.3. Test önceliklendirme

Araştırmalar test maliyetinin toplam yazılım geliştirme maliyetinin yarısından fazlasını tuttuğunu göstermektedir. Test süreci, tasarım ekibi yada testçiler tarafından oluşturulan test senaryosunun bir düzende işletilmesi sürecidir. Varsa yazılım

hatalarının tespit edilip düzeltilmesiyle yazılımın beklenen kaliteye ulaşması sağlamak amacıyla gerçekleştirilir.

Test maliyetinin yüksekliği ve insan faktörü dikkate alındığında hatasız bir yazılımdan bahsetmek mümkün değildir. Ancak test yoluyla mümkün olduğunca çok hatadan arındırılan yazılımların kalite ve güvenilirliğini artırmak mümkündür. Yazılımın kapsam genişliğine ve maliyetine göre her zaman tamamını kontol etmek yani test etmek mümkün olmayabilir. Bu gibi durumlarda planlanan test senaryosunun bir kısmı işletilerek test bütçesi dahilinde test işlemi gerçekleştirilir. Bu noktada yazılımın hangi kısımlarının öncelikle test edilmesi gerektiğine karar vermek kritik bir öneme sahiptir. Bu amaçla literatürde test durumlarını seçme (*ing. test case selection*), test takımını arıtma (*ing. test suite reduction*), test durumlarını süzme (*ing. test case filtering*) ve test durumlarını önceliklendirme (*ing. test case prioritization*) yöntemleri kullanılmaktadır (Rothermel ve Harrold, 1993; Chen vd., 1994; Wong vd., 1997; Rothermel ve Harrold, 1997; Ball, 1998; Ensan vd., 2011; Harrold vd., 1993; Wong vd., 1995; Rothermel vd., 2002; Heimdahl ve George, 2004; Fraser ve Wotawa, 2007; Masri vd., 2007; Elbaum vd., 2000; 2001, Kim ve Porter, 2002; Srivastava ve Thiagrajan, 2002; Korel vd., 2005; Fraser ve Wotawa, 2007; Acharya vd., 2010; Athira ve Samuel, 2010; Salem ve Hassan, 2010; Bryce vd., 2011; Mohanty, vd., 2011; Jiang, vd., 2011).

- *Test durumlarını seçme yöntemleri*, yazılımın mümkün olan en iyi kapsamı sağlamak koşuluyla aralarındaki benzerlikler en az olan test durumlarını seçerler. Test takımı içindeki testlerin benzerlik derecelerini belirleyerek benzerliği en yüksek olan testlerden yalnızca birini seçip diğerlerini elimine ederek test takımını küçültürler.
- *Test takımını arıtma yöntemleri,* orijinal test takımının hata yakalama etkinliğini korurken test dizilerinin sayısını azaltmaya çalışırlar. Bunu da test takımının içindeki tekrarlayan durumları elimine ederek gerçekleştirirler.
- *Test durumlarını süzme yöntemleri,* testleri benzerliklerine göre gruplayıp, her gruptan yalnızca bir testi seçmek yoluyla test takımını küçültmeyi amaçlayan yöntemlerdir.

- *Test önceliklendirme yöntemleri,* testleri bazı öncelik kriterlerine göre sıralamaya dayalı tekniklerdir. Test takımındaki tüm testleri çalıştırmak mümkün olmadığında ilk çalıştırılan testlerin hata yakalama oranları diğerlerinden daha yüksek olacak şekilde testleri sıralar. Test takımını küçültmeyi amaçlayan diğer üç yöntemden farkı ise, hiçbir test dizisini elimine etmemeleridir. Bu sayede de test bütçesinin el verdiği kadar çok testin yapılması mümkün olur.

Wong vd. (1997) ilk kez ek kapsam başına maliyetin artması kriterine göre test durumlarını sıralamaya dayalı bir yaklaşım önermişdir. Böylece testleri kapsam kriterine göre sıralayarak test maliyetinin düşürülmesi amaçlanmıştır (Wong vd., 1997). Ancak test önceliklendirme problemi ilk kez formal olarak Elbaum vd. (2001) tarafından Tablo 1.1 de verildiği gibi tanımlanmıştır.

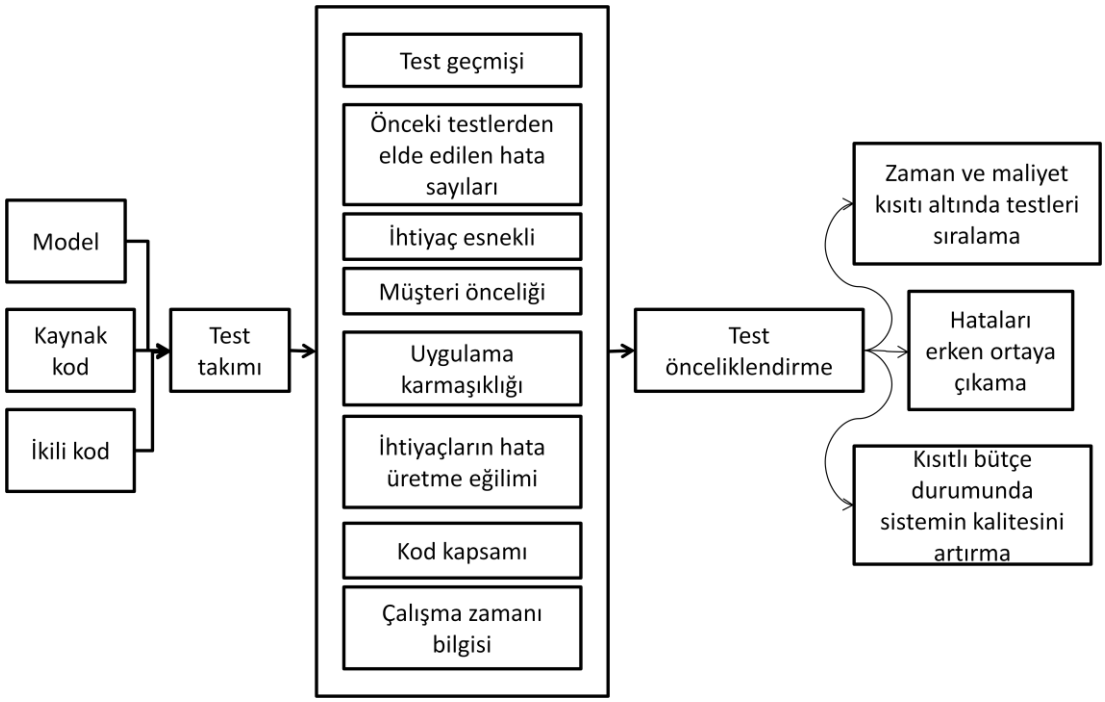
Tablo 1.1. Test önceliklendirme probleminin tanımı

Koşullar: *T, test takımı,* *PT, T'nin permütasyonlarının kümesi* *f, PT den Reel sayılara bir fonksiyon olmak üzere*
Problem: *Öyle* $T' \in PT$ *bul ki* $(\forall T'')$ $(T'' \neq T')$ $[f(T') \geq f(T'')]$ *olsun.*

Burada T, test senaryoları, PT, test senaryolarının farklı sıralamalarından oluşan dizisi ve f, PT den reel sayılara bir amaç fonksiyonu olmak üzere öyle bir test takımı seçilmelidir ki; bu test takımının önceliği PT içindeki diğer test takımlarının önceliğinden daha yüksek olmalıdır.

Test önceliklendirme problemini çözmek için literatürde önerilmiş çalışmaların büyük çoğunluğu resgreyon testi sürecinde kullanılmaktadır (Elbaum vd., 2000; Jones ve Harrold, 2001; Kim ve Porter, 2002; Qu vd., 2007; Athira ve Samuel, 2010). Regresyon testleri yazılımda değişiklik yapılması durumunda yapılan bu değişikliklerin sistemde bir bozulmaya neden olup olmadığını test etmek amacıyla yapılan testlerin genel ismidir. Regresyon testleri daha çok kullanıma sunulmuş yazılımların bakımı ve güncelleştirmesi aşamasında kullanılırlar. Mevcut yaklaşımların büyük çoğunluğu programın kaynak kodunu kullanılarak veya ikili kodları karşılaştırılarak test önceliklerini belirlemektedir. Şekil 1.3'de mevcut test önceliklendirme süreçlerinin izlediği yol ve sonuçları genellenmiştir. Bu yaklaşımlar model, kaynak kod veya ikili kodlar kullanılarak üretilen test takımını, test geçmişine

ait bilgiler, kod kapsamı veya müşteri öncelikleri gibi sisteme ilişkin elde olan bilgileri kullanılarak önceliklendirirler. Amaçları zaman ve maliyet kısıtı altında bir yazılımın kalitesini artırmak ve varsa sistemdeki hataları mümkün olduğunca erken ortaya çıkarmaktır.



Şekil 1.3. Mevcut test süreci önceliklendirme tekniklerine genel bakış

Elbaum vd. regresyon testi sürecinde test önceliklendirme konusunda çok sayıda çalışma yapmışlardır (Elbaum, vd., 2000; Elbaum vd., 2001; Elbaum vd., 2004). Bu çalışmalarda yazılımın önceki kullanımlarına yada önceki testlerine dair bilgiler mevcuttur ve test dizilerinin önceliklendirilmesinde bu bilgilere başvurulur. Kim ve Porter test önceliklendirme için test dizilerinin daha önceki çalıştırılma bilgilerine dayanan bir yöntem önermişlerdir (Kim ve Porter, 2002). Sirvastava ve Thiagarajan test takımını optimize etmek amacıyla Echelon olarak adlandırılan bir sistem geliştirmişlerdir (Srivastava ve Thiagarajan, 2002). Srikanth vd. (2005) test gereksinimlerine dayalı PORT (*ing. Prioritization of Requirements for Test*) olarak adlandırdıkları bir test önceliklendirme metodu önermişlerdir. Bu metotda testleri önceliklendirmek için dört faktör dikkate alınmıştır: ihtiyaç esnekliği (*ing. requirement volatility*), müşteri önceliği (*ing. customer priority*), uygulama karmaşıklığı (*ing. implementation complexity*) ve ihtiyaçların hata üretme eğilimi

(*ing. fault proneness of the requirements*). Krishnamoorthi ve Sahaaya Arul Mary hem yeni test durumları hem de regresyon testleri için kullanılabilen bir gereksinim tabanlı test önceliklendirme yaklaşımı önermiştir (Krishnamoorthi ve Sahaaya Arul Mary, 2009).

Walcott vd. (2006) zaman kısıtı altında genetik algoritma kullanarak test takımındaki testleri regresyon testler için yeniden sıralamayı sağlayan bir test önceliklendirme tekniği geliştirmişlerdir. Jeffrey ve Gupta (2006) programda yapılan bir değişikliğin regresyon test takımındaki bir testin sonucunu etkiliyorsa bu testin önce test edilmesi gerektiğini öneren yeni bir test önceliklendirme yaklaşımı önermiştir. Bryce ve Memon (2007) kapsamlar arası etkileşimlere dayanan etkileşim tabanlı (*ing. interaction-based*) önceliklendirme metodunu önermiştir. Ma ve Zhao (2008) program yapısının analizine dayalı bir test öncelliklendirme yaklaşımı önermiştir.

Ledru vd. (2009) koda ihtiyaç duymadan test yapma imkanı sağlayan, test metnindeki diziler (*veya stringler)* arasındaki uzaklıklara dayanan bir önceliklendirme metodu önermişlerdir. Korel vd. (2005) sistem modeli üzerinden kaynak kodlara ihtiyaç duymayan model tabanlı test önceliklendirme yaklaşımı önermiştir. Bu yaklaşım orjinal sistem modelini kullanarak, değiştirilmiş sistemle orjinal sistem arasındaki farklara dayalı bir önceliklendirme yapar. Regresyon testleri kapsamındadır. Bu çalışmada modelleme için durum tabanlı (*ing: state-based*) bir yapı kullanılmıştır.

1.2.4. Sistem modelleme

Test süreci önceliklerini belirleme konusunda kullanılan mevcut tekniklerin çoğu test edilen sistemin önceki çalıştırılmalarına ait hata sayılarına, çalıştırma zaman bilgisine veya programın kaynak kodlarına ihtiyaç duymaktadır. Ancak yazılımın kaynak koduna ulaşılamadığında ve yazılımın daha önceki çalıştırılma bilgilerinin de bulunmadığı durumlarda mevcut yaklaşımlar yetersiz kalmaktadır. Yazılım firmaları için program kodlarını açıklamak gizlilik gereği çoğu kez mümkün olmaz. Bu durumda program kodlarına ihtiyaç duymadan sistem modeli üzerinden test işleminin yürütülebilmesine olanak sağlayan model tabanlı test yaklaşımları kullanılmaktadır

(Broy vd., 2005). Sistem modeli, kullanıcı ihtiyaçlarını karşılayacak şekilde sistem davranışlarını gösteren bir yapıdır. Bu amaçla geliştirilmiş yazılım modelleme teknikleri şunlardır: Durum Şemaları (*ing. State Charts*), Sonlu Durum Makineleri (*ing. Finite State Machines*), Yönerge Tanımlama Dili (*ing. Specification Description Language*) ve Olay Dizisi Grafikleri (ODG) (Belli, 2001; Harel, 1987; Carroll ve Long, 1989; Cheng ve Krishnakumar, 1993; Dssouli vd., 1999). Ayrıca bu modelleme teknikleri kullanılarak model tabanlı test üretme algoritmaları geliştirilmiştir (*ing. model based test generation*) (Dick ve Faivre, 1992; Cheng ve Krishnakumar, 1993; Jourdan vd., 2006).

Bu çalışmada sistemin işleyişinin gözlemlenmesiyle modelleme imkânı verdiği ve sistem kodlarına ihtiyaç duymadığı için ODG ile modelleme tercih edilmiştir. ODG yaklaşımı, sistem davranışlarını ve kullanıcı hareketlerini "*olay*" olarak görür ve sistemin kullanıcı beklentileri doğrultusundaki hareketleri arzu edilen olayları, beklenmedik davranışlarıda arzu edilmeyen olayları temsil eder. Sistemin beklenmedik davranışları "*hata*" olarak adlandırılır.

1.2.5. Test takımı üretme

Test işlemi bir test senaryosu üzerinden yürütülür. Test takımı olarak adlandırılan bu senaryonun önceden oluşturulması gerekir. Literatürde test sürecinde kullanılan modelleme türüne ve uygulanan test tekniğine göre farklı test takımı üretme algoritmaları mevcuttur (Dick ve Faivre, 1992; Cheng ve Krishnakumar, 1993; Offutt ve Abdurrazik, 1999; Edvardson, 1999; Jourdan vd., 2006). Kara kutu test yöntemleri altında incelenen model tabanlı test yaklaşımında ODG ile geliştirilen model üzerinden, kesikli olayların sonlu bir kümesinden meydana gelen bir test takımı üretilir. Test takımı çok sayıda test dizisinden oluşur. Test süreci de test takımının işletilmesini ifade eder. Ancak tüm test dizilerinin işletilmesi mümkün olmadığı zaman, hangi testlerin önce işletileceğine karar vermek gerekir. Yazılım kalitesi ve güvenilirliği açısından da önce işletilecek testlerin sistemdeki hataları ortaya çıkarma konusunda diğer testlerden daha iyi olması beklenir. TÖP'i çözmek için önerilmiş mevcut yaklaşımlar söz konusu modeldeki tüm ilişki çiftlerini kapsayan minimum maliyetli test takımını kullanmaktadırlar (Elbaum vd., 2002; Bryce ve Colbourn, 2006). TÖP Çinli Postacı Probleminin (*ing. Chinese Postman*

Problem (CPP)) genelleştirilmiş bir durumu olarak ele alınır (Edmonds ve Johnson, 1973; Belli, 2001; Belli vd., 2006; Belli ve Budnik, 2007). Bu çalışmada bir ODG'de her arktan en az bir kez geçilerek yani her olay çifti en az bir kez kullanılarak en kısa turun oluşturulmasını amaçlayan ÇPP kullanılarak üretilen test takımı üzerinden test işlemi gerçekleştirilmektedir.

1.2.6. Kümeleme analizi

Kümeleme analizi (*ing. cluster analysis*), çok boyutlu uzayda yer alan bir veri seti içindeki yapılandırılmamış verilerin benzerliklerini veya farklılıklarını dikkate alarak alt gruplar oluşturmak için geliştirilen çok değişkenli veri analiz yöntemlerinden biridir (Anderberg, 1973; Aldenderfer ve Blashfield, 1985). Kümeleme yöntemlerinin amacı verileri benzerliklerine göre bir araya getirip mümkün olduğunca kendi içinde homojen ve birbirleri arasında heterojen gruplar oluşturmaktır (Hoppner vd., 1999; Tatlıdil, 2002). Literatürde kümeleme analizi başlığı altında çok sayıda kümeleme algoritması bulunmaktadır (Tan vd., 2006). Bu kümeleme algoritmaları verilere, kullandığı kümeleme kriterlerine ve dayandığı teorilere göre farklı şekillerde sınıflandırılabilirler (Jain vd., 1999; Halkidi vd., 2001). Hiyerarşik (*ing. hierarchical*), yoğunluğa dayalı (*ing. density based*), grid tabanlı (*ing. grid-based*), model tabanlı (*ing. model-based*) ve bölünmeli (*ing. partitional*) kümeleme gibi farklı kümeleme yaklaşımları mevcuttur. Bu çalışmada bölünmeli kümeleme yöntemleri kullanılacaktır.

Bölünmeli kümeleme yöntemleri, *c* giriş parametresini alarak *n* tane nesneyi *c* tane kümeye böler (Jain vd., 1999). Bu yöntemler, başlangıç küme merkezlerini rasgele belirleyerek her bir nesnenin / birimin bu küme merkezine olan uzaklığına göre yeni küme merkezlerini oluşturur. Bu işlem kendi içlerinde homojen ve birbirleri arasından mümkün olduğunca heterojen *c* adet küme oluşturuluncaya kadar döngüsel olarak tekrarlanır. Bölünmeli kümeleme yöntemlerinden en bilineni istatistiksel kümeleme yöntemi olan k-ortalamalar (*ing. k-means*)'dır. Merkez noktanın kümeyi temsil etmesi esasına dayanır ve eşit büyüklükte küresel kümeler bulma eğilimindedir (Han ve Kamber, 2001).

Klasik metotlar olarak bilenen istatistiksel metotlar bazı varsayımlara dayanırlar ve bu varsayımların sağlanmadığı durumlarda yetersiz kalabilirler. Bu nedenle istatistiksel metotlara alternatif olarak insan beyninin düşünce sürecini modelleyerek problemlere çözümler üretmeyi amaçlayan ve yumuşak hesaplama (*ing. soft computing*) yöntemleri geliştirilmiştir (Zadeh, 1994; Bonissone, 1997; Jang, vd., 1997).

1.2.7. Yumuşak hesaplamaya dayalı kümeleme

Geleneksel – klasik hesaplama - metotları kesinlik, doğruluk, tamlık gibi bazı kısıtlara sahiptir. Belirsizlik, eksik veri, kısmi doğruluk gibi muğlâk durumlarda kullanılamayabilirler. Oysa bulanık mantık, genetik algoritmalar, yapay sinir ağları, gibi yumuşak hesaplama yöntemleri insanın düşünce sürecini modelleyerek problemlere çözüm aradıkları için kesin olmama, belirsizlik, kısmi doğruluk gibi durumlarda başarılı sonuçlar üretebilmektedir (Zadeh, 1965; 1994; 1996; Jang vd., 1997; Holland, 1975; Goldberg, 1989; Haykin, 1994). Bu nedenle son yıllarda pek çok uygulamada klasik (istatistiksel) metotlara alternatif olarak tercih edilmektedirler. Bu çalışmaların sonuçları da yumaşak hesaplama yöntemlerinin gerçek problemlere düşük maliyetli ve yaklaşık çözümler bulmakta başarılı olduğunu göstermiştir.

Bu çalışmada belirsizlik içeren pratikte ortaya çıkan problemlere düşük maliyetli ve yaklaşık çözümler bulmak için klasik metotlara alternatif olarak kullanılan yumuşak hesaplama metotları tercih edilmiştir. Yapay sinir ağ tabanlı danışmansız kümeleme algoritmalarından yarışmacı öğrenme (YÖ) (*ing. competitive learning, CL*) algoritması ve bulanık mantık tabanlı bulanık c-ortalamalar (BCO) (*ing: fuzzy c-means*) algoritması kullanılarak kümeleme yapılmıştır. Önerilen test önceliklendirme metodunu tanıtmak için tercih edilmiş olan bu iki algoritma klasik ve bulanık kümeleme algoritmalarının en çok kullanılan ve işlem karmaşıklığı en düşük olan örnekleridir.

1.2.7.1. Klasik kümeleme – yapay sinir ağları

Yapay sinir ağları (YSA) paralel çalışan, doğrusal olmayan ve biyolojik sinir ağlarını taklit eden işlem birimlerinden oluşan bilgisayar sistemleridir. İlk YSA modeli 1943 yılında McCulloch ve Pitts tarafından geliştirilmiştir (McCulloch ve Pitts, 1943). 1948'de Wiener, ilk kez yapay sinir hücrelerinin çalışma prensiplerini ve davranış özelliklerini tanımlamıştır (Wiener, 1948). 1949'da ise Hebb "Davranış Organizasyonu (*ing. Organization of Behavior*)" adlı çalışmasında ilk kez sinir ağlarının öğrenebileceğine dair teoriyi ortaya atmıştır (Hebb, 1949). 1959 da geliştirilen ADALINE (*ing. Adaptive Linear Neuron or later Adaptive Linear Element*) modeli YSA'ların mühendislik uygulamaları için başlangıç kabul edilmiştir (Widrof ve Hoff, 1959). Uzak mesafelere sahip telefon hatlarındaki gürültü ve yankıları yok eden bir adaptif filtre olarak kullanılan bu model gerçek dünya problemlerine uygulanan ilk YSA modeli olmuştur.

YSA'lar eğitim algoritmaları kullanılarak eğitilirler (Haykin, 1994). Danışmanlı (*ing. supervised learning*) ve danışmansız öğrenme (*ing. unsupervised learning*) (Barlow, 1989) olmak üzere iki tür eğitim algoritması kullanırlar. Ağa verilen girdi bilgilerine karşılık çıktı bilgileri mevcut ise eğitme algoritması olarak danışmanlı öğrenme algoritmaları kullanılır. Ağın çıktıları ile beklenen çıktılar karşılaştırılır ve aradaki fark belli bir hata düzeyine ininceye kadar eğitim boyunca ağırlıklar düzenlenir. Verilen girdi bilgilerine karşılık beklenen çıktı bilgilerinin mevcut olmadığı durumlarda danışmansız öğrenme algoritmaları tercih edilir.

1970'lerin başlarında Kohonen öğrenme ve bileşik hafızalar üzerine çalışmalar yapmış ve bu çalışmalar danışmansız öğrenme algoritmalarının temelini oluşturmuştur. Willshaw ve Malsburg 1976'da özdenetimli harita (kendi kendini düzenlemeli özellik haritaları) (*ing. self organized feature maps*) olarak adlandırılan kümeleme algoritmasını geliştirmişlerdir. Kohonen 1982 ve 1984'deki çalışmalarıyla özdenetimli haritalar kuramını beyindeki sinir ağlarının karşılaştırmalı haritasını çıkaracak şekilde geliştirmiş ve bu model daha sonraki çalışmalarda yüksek boyutlu girdilerin daha düşük boyutlu çıktılarla temsil edilmesi amacıyla kullanılmaya başlanmıştır. Çok boyutlu verilerin görsel olarak ifade edilmesinde başarılı sonuçlar veren bu model literatürde Kohonen ağları (*ing. Kohonen Networks*) olarak da adlandırılmaktadır (Kohonen, 1982; Kohonen, 1984). Kohonen ağları girdi

vektörlerinin dağılımlarını öğrenebilme ve girdi vektörlerini sınıflandırabilme yeteneklerinden dolayı sınıflandırma ve kümeleme amacıyla kullanılmaktadırlar (Melin ve Castillo, 2005; Xu vd., 1993). Ayrıca bu ağlar literatürde kümelemeye ek olarak, sürekli fonksiyonların boyutlarının düşürülmesinde veya kesikli (*ing. discrete*) hale getirilmesinde ve veri içindeki özelliklerin ortaya çıkarılması amacıyla kullanılmaktadırlar (Grossberg, 1987; Chou vd., 1995; Ahalt vd., 1990).

Kohonen ağları girdi ve çıktı katmanı olmak üzere iki katmandan oluşurlar (Fu,1994). Girdi katmanı ve yarışmacı katman olarak da bilinen çıktı katmanındaki tüm işlemci elemanları ağırlık vektörleri ile birbirlerine bağlantılıdır. Bu ağırlık vektörlerinin değerlerinin belirlenme süreci eğitim süreci olarak bilinmektedir. Geleneksel hesaplama yöntemlerinden farklı olarak, örnek olaylardan yola çıkarak genelleme yapabilme, öğrenme yoluyla yeni bilgiler üretebilme ve hiç görmediği örneklerle karşılaştığında deneyimlerden yararlanarak karar verebilme yeteneğine sahiptirler. YSA'lar sınıflandırma, modelleme ve tahmin olmak üzere pek çok alana başarıyla uygulanmaktadırlar (Kohonen, 1989; Fu, 1994). YSA'lar kendisine verilen örnekler arasındaki ilişkileri eğitim yoluyla ortaya çıkarabilme ve bu örnekleri kümeleyebilme yeteneğine sahip olmalarından dolayı kümeleme amacıyla kullanılabilmektedirler.

Klasik kümele problemlerinin çözümünde tercih edilen yarışmacı öğrenme ağlarının eğitiminde kullanılan öğrenme kuralı ilk kez Rummelhart ve Zipser tarafından tanımlanmıştır. Bu kurala göre küme merkezlerine karşılık gelen ağırlık vektörleri kendi aralarında yarışırlar ve bakılan anda yalnızca bir ağırlık vektörü eğitilir. Kazanan girdi vektörüne en yakın olan ağırlık vektörüdür. Bundan dolayı yarışmacı öğrenme algoritmasının eğitim stratejisi "Kazanan Herşeyi Alır" (*ing. Winner-Takes-all*) olarak adlandırılmaktadır (Rummelhart ve Zipser, 1985; Fu, 1994; Haykin, 1994). Yani eğitim süresi boyunca her adımda yalnızca bir ağırlık merkezi eğitilir. Eğitim süreci tüm ağırlık merkezleri eğitilene kadar devam eder. Ağın eğitimi tamalandıktan sonra klasik kümeleme mantığı kullanıralarak birimler sınıflandırılır. Böylece birimlerin / olayların nihai kümeleri elde edilir.

1.2.7.2. Bulanık kümeleme – bulanık c-ortalamalar

Aristo mantığı olarak da bilinen klasik kümeleme {0,1} ikili mantığa dayalıdır. Klasik kümeleme de kesinlik vardır, bir nesne bir kümenin ya elemanıdır yada değildir. Oysa 1965 yılında Zadeh tarafından tanımlanan bulanık mantığa göre bir nesne aynı anda birden fazla kümenin elemanı olabilir ve [0, 1] aralığında bütün değerleri alabilir (Zadeh, 1965).

Bulanık kümeleme algoritmalarından en popüler olanı Dunn tarafından önerilen ve Bezdek tarafından geliştirilen bulanık c-ortalamalar (BCO) (*ing. Fuzzy c-means, FCM*) algoritmasıdır (Dunn, 1973; Bezdek, 1981). BCO algoritması, verilerin sıfır ile bir arasında farklı üyelik değerleriyle (*ing. membership degrees*) bir veya daha fazla kümeye ait olabilmesi mantığına dayanan bulanık kümeleme algoritmasıdır. BCO algoritması küme içindeki farklılığı ölçmek için amaç fonksiyonu kullanır ve bu amaç fonksiyonunu minimize ederek en iyi bölünmeyi elde etmeye çalışır (Bezdek, 1981). Bu tarz kümeleme yöntemlerinde bir eleman aynı anda birden fazla kümeye ait olabilir ancak en yüksek üyelik değeri ile ait olduğu kümeye atanarak sınıflandırılır. Klasik kümelemeye göre daha esnektir. Bu nedenle gerçek dünyaya yönelik uygulamalarda gruplar arası sınırların kesin olmadığı durumlarda bulanık kümeleme daha iyi sonuçlar üretmektedir. Bulanık kümeleme analizi, örüntü tanıma, görüntü işleme, bulanık modelleme gibi alanlarda sıklıkla kullanılmaktadır (Zadeh, 1996; Hoppner vd., 1999).

1.2.8. Kümeleme tabanlı test önceliklendirme

2006 yılından bu yana devam eden bu tez çalışmasında kümeleme tabanlı test önceliklendirme yöntemi önerilmiş ve yöntem gelişim süreci boyunca uluslararası bilimsel toplantılarda tartışılmış ve bazı bilimsel dergilerde bildiri ve makale olarak yayınlanmıştır. Kronolojik sıraya göre çalışmalar şu şekilde gelişmiştir. Kümelemeye dayalı model tabanlı test önceliklendirme yöntemi ilk olarak Gökçe vd., (2006) çalışmasında tanımlanmış ve test önceliklendirme problemlerinin çözümü için yapay sinir ağ tabanlı bir kümeleme algoritması olan yarışmacı öğrenme algoritmasının kullanılabileceği gösterilmiştir. Belli vd. (2007a; 2007b) çalışmalarında olayları nitelendiren yeni faktörler tanımlanarak yöntem geliştirilmiş ve bulanık mantığa

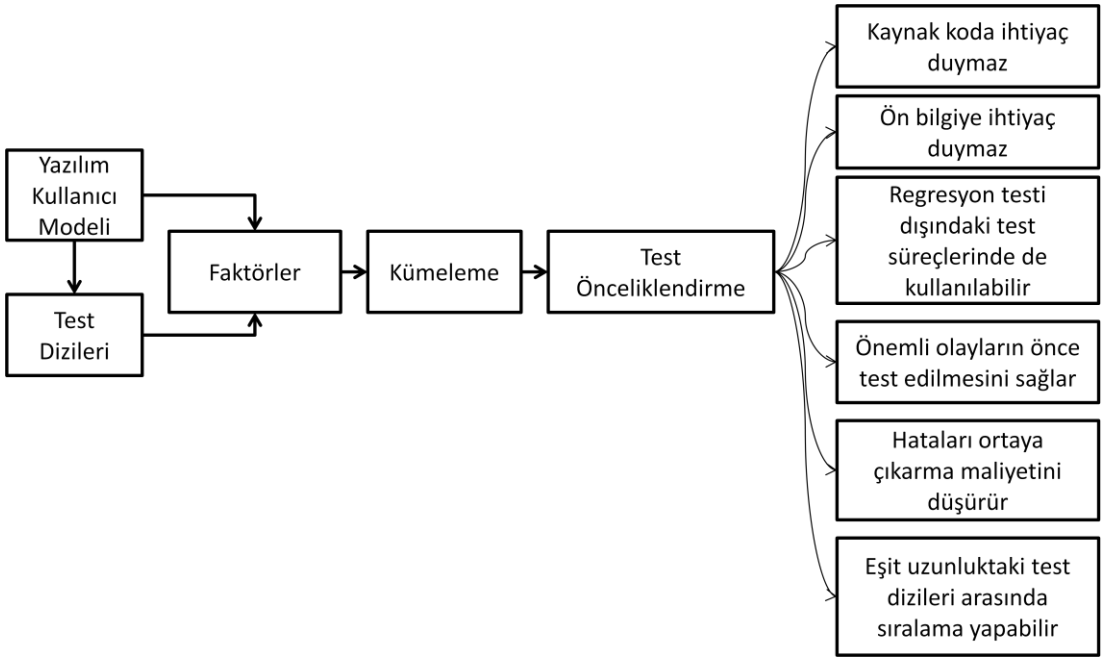
dayalı kümeleme algoritması ilk kez bu çalışmada test önceliklendirme amacıyla kullanılmıştır. Yapay sinir ağları ve bulanık kümeleme algoritmalarının önceliklendirme açısından karşılaştırması Belli vd. (2009) çalışmasında verilmiştir. Ayrıca test önceliklendirme metodunu farklı düzeylerde modellenmiş sistemlerin testine de uygulanmıştır (Belli ve Gökçe, 2010). Yine söz konusu yönteme ilişkin bir başka uygulama da Belli vd. (2011) çalışmasında gerçekleştirilmiştir.

Bu tez çalışmasıyla sisteme ilişkin hiçbir ön bilgi bulunmadığı durumda yalnızca sistem modeli üzerinden önceliklendirmeye imkan sağlayan yeni yöntem önerilmiştir. Bu yöntemin gelişim aşamaları açıklanarak örnek uygulamalar gerçekleştirilmiştir. Ayrıca önerilen yöntemin performansını değerlendirmek için test sonucunda elde edilen hata bilgisine dayanan değerlendirme kriterleri önerilmiştir. Bu kriterler yardımıyla kümeleme ile test önceliklendirme yaklaşımının yazılım hatalarını ortaya çıkarmadaki başarımı tartışılmıştır.

1.3. Tezin Katkısı

Mevcut test süreci önceliklendirme yöntemlerinin büyük bir çoğunluğu regresyon testi sürecinde kullanılmaktadır. Regreyon testleri sistemin önceki çalıştırılmalarına ait hata bilgilerini, çalıştırılma zaman bilgileri gibi bazı ön bilgilere ihtiyaç duyarlar. Bu bilgileri kullanarak test dizilerini sıralarlar. Ancak ön bilgi olmadığı durumda bu yaklaşımlar kullanılamaz. Bu çalışmanın katkısı ve mevcut çalışmalardan en önemli farkı kaynak kod ve sisteme ait hiçbir bilgi olmadığında yalnızca sistem modelini kullanarak test dizilerini sıralamaya imkan sağlayan bir yöntem önermesidir. Kümeleme ile test önceliklendirme yöntemi sistem modelini oluşturan bileşenlerin kümeleme yoluyla ağırlıklandırılması ve bu şekilde önemli bileşenleri içeren test dizilerinin önce test edilmesini önermektedir. Literatürde test takımını filtreleme / süzmek için daha önce kümeleme algoritmaları kullanılmıştır. Ancak bu yaklaşımlar test dizilerini mevcut bilgilere (test geçmişine ve hata sayılarına dair ön bilgi mevcut) göre kümelendirir ve her kümeden belirli sayıda test dizisinin test edilmesini öngörür. Bu şekilde test takımını küçülterek test maliyetini düşürmek için kullanımıştır. Bu çalışmada, kümeleme yaklaşımı ilk kez test önceliklendirme sürecinde kullanılmış ve test dizilerini oluşturan olaylar kümelenmiştir. Olayların

önem grupları belirlenerek test dizileri hata yakalama performansı yüksek olanlardan başlamak üzere sıralanmıştır. Ve hiçbir test dizisi elimine edilmemiştir.



Şekil 1.4. Önerilen yöntemin farkı ve avantajları

Kümeleme ile test önceliklendirme yönteminin mevcut yöntemlerden farkı ve avantajları Şekil 1.4'de ana hatları ile verilmiştir. Sistemin kullanıcı modeli ve bu model üzerinden üretilen test dizileri kullanılarak olayları nitelendirmek için çeşitli faktörler tanımlanmıştır. Bu faktörler sistem için önemli olayları veya hata düzeltme maliyeti yüksek olayları belirlemek amacıyla kullanılmıştır. Bu faktörlerin hiç birisi ön bilgi olarak sistemin önceki çalıştırmalarına ait hata sayısını veya kod kapsamı bilgilerini kullanmaz. Faktörlerin olaylara ait değerleri olayların modeldeki ve test dizilerindeki kullanımlarına göre atanır. Faktörlerin değerlerinin büyük olması söz konusu olayın sistem içinde sık kullanıldığını veya hata ortaya çıkarma potansiyelinin yüksek olduğunu gösterir. Söz konusu faktörlerle ağırlıklandırılan olayların önem grupları kümeleme yöntemleriyle belirlenmiştir. Yumuşak hesaplama yöntemlerinden, yapay sinir ağları (*ing. neural networks*) ve bulanık kümeleme (*ing. fuzzy clustering*) yaklaşımları olayları kümelemek için kullanılmıştır. Yüksek değerlere sahip olaylar en önemli olaylar olacak şekilde kümelere önemlilik değerleri atanmıştır. Nihai test takımı yüksek önemliliğe sahip olayları içeren test dizileri sıralanarak elde edilmiştir.

Önerilen yöntem yazılımın kaynak koduna sahip olunmadığında ve yazılımın daha önceki çalıştırlmalarına ait her hangi bir bilgi bulunmadığı durumda test dizilerinin sıralanmasını sağlar. Regresyon testleri dışındaki test süreçlerinde de kullanılabilir. Test dizisinin uzunluğundan çok diziyi oluşturan olayların önemini ortaya çıkarmayı amaçladığı için eşit uzunluktaki test dizileri arasında öncelik açısından ayrım yapabilir. Hata yakalama potansiyeli yüksek olayların önce test edilmesini önerdiğinden düşük maliyetle yazılım kalitesini artırabilir.

1.4. Tezin Organizasyonu

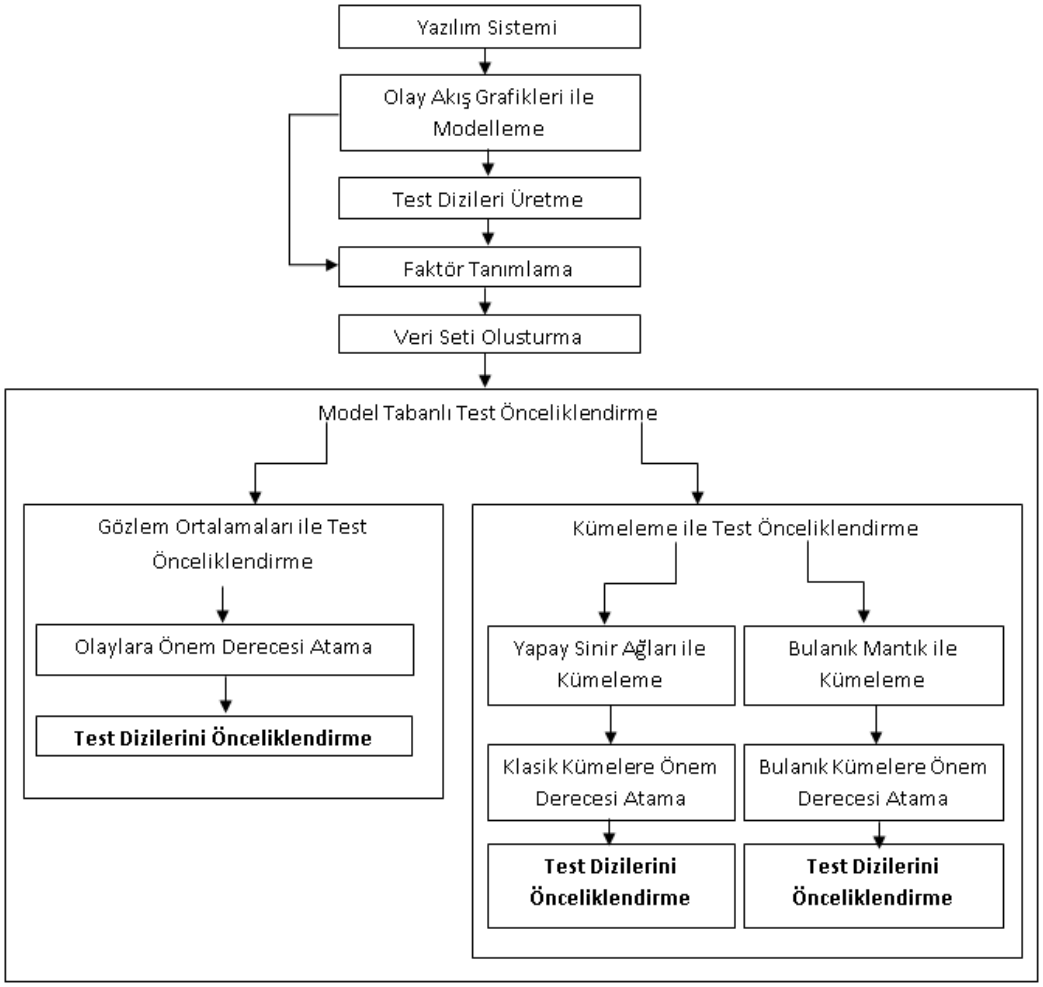
Bu tezin organisayonu şu şekildedir. Bölüm 2'de olay dizisi grafikleri ile yazılım modellemesi ve kümeleme tabanlı test önceliklendirme metodu açıklanmıştır. Bölüm 3'de önerilen kümeleme ile test önceliklendirme yönteminin Web tabanlı bir yazılım sisteminin (ISELTA turizm acentaları için geliştirilmiş yazılım portalı) test sürecine uygulanışı verilerek bulgular irdelenmiştir. Bölüm 4'de sonuçlar tartışılmış ve Bölüm 5'de genel sonuçlar verilerek gelecekte yapılabilecek olası çalışmalar sunulmuştur.

2. MALZEME VE YÖNTEM

Bu bölümde yazılım sisteminin modellenmesinde kullanılan olay dizisi grafikleri kısaca tanımlanacak ve kümeleme tabanlı test önceliklendirme metodu açıklanacaktır. Metodun anlatılması sırasında izlenen konu akışı Şekil 2.1 de verilmiştir.

Test önceliklendirme metotları, sistemdeki hataları mümkün olduğunca erken tespit ederek sistemi hatalarından arındırmak ve dolayısıyla yazılımın kalite ve güvenilirliğini artırmayı amaçlayan yöntemlerdir. Bu yolla sıralanmış bir test takımından beklenen, ilk sıradaki test dizilerinin hata ortaya çıkarma oranlarının sonraki sıralardaki testlerden daha yüksek olmasıdır.

Bu çalışmada sistem bileşenlerinin kümelenmesine dayalı yeni bir test önceliklendirme yöntemi önerilmektedir. Bu amaçla, test dizilerini oluşturan olaylar için, hatayı erken ortaya çıkarabilme kapasiteleri ya da test dizilerinin hata düzeltme maliyetlerini ortaya koyabilecek 13 faktör tanımlanmıştır. Tanımlanan faktörlerin hiçbiri yazılım koduna ya da yazılımın önceki çalıştırma bilgilerine ihtiyaç duymamaktadır. Söz konusu faktörler olayların kümelenmesi amacıyla kullanılmıştır. Olay kümeleri birbiriyle kıyaslanarak önemlilik değerlerine göre ağırlıklandırılmıştır. Test dizileri içerdikleri olayların önemliliği doğrultusunda önceliklendirilerek sıralanmıştır. Mevcut yaklaşımlar sıralama kriteri olarak kapsam ve maliyet kriterini kullanmaktadır. Ancak bu kriterler test dizilerinin uzunluğuna bağlıdır ve kısa testleri seçme eğilimindedir. Dahası eşit uzunluktaki test dizileri arasında öncelik açısından ayrım yapamayabilirler. Kümeleme ile test önceliklendirme yöntemi test dizilerinin uzunluğuna olan bağımlılık ortadan kaldırarak diziyi oluşturan olayların önemini ortaya çıkamıştır. Bu sayede eşit uzunluktaki test dizilerini farklı öncelik değerlerine göre sıralamak mümkün olmuştur.

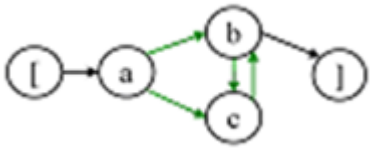


Şekil 2.1. Kümeleme ile model tabanlı test önceliklendirme yöntemi uygulama süreci

2.1. Modelleme - Olay Dizisi Grafikleri

Kara kutu test tekniklerinde test takımını önceliklendirmek için sistem modeline dayalı test yaklaşımları kullanılmaktadır. Bu çalışmada sistemi modellemek için Olay dizisi grafikleri (ODG) tercih edilmiştir. ODG, kaynak koda ihtiyaç duymadan sistem bileşenlerini (menuler, onay düğmeleri, pencereler, işaret kutuları vs.) ve bu bileşenler arasındaki geçişleri grafiksel olarak modellemek için kullanılırlar.

Bir yazılım sisteminde "olay" olarak adlandırılan sistem bileşenlerinin her biri çevresel veya kullanıcı kaynaklı uyarıcılar olabileceği gibi sistem aktivitesinin farklı aşamalarını içeren dıştan gözlemlenebilen olgulardır (Belli, 2001; Belli ve Budnik, 2007). ODG, matematiksel olarak yönlü ve etiketlenmiş bir graftır. Şekil 2.2'de üç tepeli basit bir ODG verilmiştir.



Şekil 2.2. Basit bir ODG (Budnik, 2006)

Formal olarak bir ODG şu şekilde tanımlanmıştır (Budnik, 2006): $ODG = (V, E)$ biçiminde düzenlenmiş olay çiftleridir. Burada *V* olaylar (*ing. node, vertex*) kümesi (yani, bir graftaki tepeler), E ise olaylar arasındaki geçişleri (*ing. arc, edge*) (yani, tepeler arasındaki yönlü arkları) temsil eder. ODG'nın *V* ile temsil edilen tepeleri bu çalışma boyunca *olay* olarak adlandırılacaktır. *V* üzerinde *a* ve *b* iki olay olarak verilsin, *a*'dan *b*'ye yönlendirilmiş bir *ab* arkı *b* olayının *a*'yı izlediğini ifade eder ve *ab* "*olay çifti (OÇ)"* (*ing. Event Pair, EP*) olarak adlandırılır. Bir olay çifti (*OÇ*) uzunluğu 2 olan bir olay dizisidir.

Bir ODG'deki tüm olay çiftleri kümesi Σ ile ifade edilir. Σ de verilen çiftler haricinde kalan olay çiftleri sistemin beklenmeyen davranışlarını ifade eder. Örneğin Şekil 2.2'de, *b* den *a* ya yönlendirilmiş bir *ba* arkı bulunmamaktadır. Ancak modelde temsil edilmeyen bu davranış sistemin kullanımı sırasında bir hata olarak ortaya çıkabilir. Bu nedenle, sistemin olası hatalı hareketlerini de dikkate almak gerekir. Dolayısıyla *ba* "*hatalı olay çifti*"*(HOÇ)* (*ing. faulty event pairs, FEP*)" olarak tanımlanır ve modelde gösterilmeyen tüm hatalı olay çiftleri yazılımda meydana gelmesi olası hatalar kümesini oluşturur (Budnik, 2006).

Bir ODG de başlangıç tepesi "[" giriş olayını ve bir final tepesi "]" çıkış olayını göstermek üzere tahsis edilmişlerdir ve Σ kümesinde yer almazlar. "[" ve "]" olayları gerçek birer olaya karşılık gelmezler, testin başlangıç ve bitişini temsil eden geleneksel gösterim biçimleridir. Bu olaylar sözde olay (*ing. pseudo node*) olarak adlandırılırken onlarla doğrudan bağlantılı arklar da sözde ark (*ing. pseudo arc*) olarak adlandırılırlar (Belli, 2001; Belli ve Budnik, 2007).

Bir "*olay dizisi (OD)*" (*ing. event sequence, ES*) en genel biçimde *n* tane ark'dan oluşan *n+1* lik ardışık olayların dizisi olarak tanımlanır ve *n+1* uzunluğundadır denir. Bir OD, ODG'nın başlangıç olayı ("[") ile başlayıp final olayında ("]") son bulursa tamdır; bu durumda dizi "*tam olay dizisi" (TOD)* (*ing. complete event*

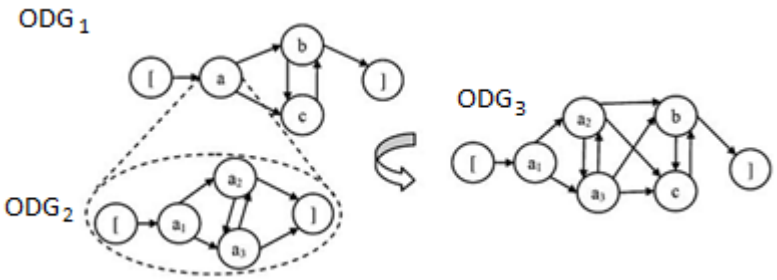
sequence, CES) olarak adlandırılır (Budnik, 2006). Literatürde TOD aynı zamanda yürüyüş (*ing. walks*) veya yol (*ing. paths*) olarak da anılmaktadır.

Bir yazılımın ODG ile modellenmesinde "*parçala ve fethet"* prensibi kullanılmaktadır. Bir sistemi tüm ayrıntılarıyla modellemek hem modelleme teknikleri hem de test açısından oldukça güçtür. Bir ODG'yi oluşturan olay sayısı arttıkça OÇ ve HOÇ sayısı artacağından, olası test dizilerinin sayısı ve dolayısıyla test maliyeti artacaktır. Bu nedenle sistemler işleyiş açısından anlamlı olacak şekilde, mümkün olan en küçük ve en basit parçalara bölünerek modellenir. Bu parçalar yazılımın modülleri, menüleri yada farklı bileşenleri olabilir. Yazılımın küçük parçalara ayırarak farklı düzeylerde modellemek, incelenecek bazı bağlantıların ihmal edilmesi anlamına gelse de sıklıkla tercih edilen bir yoldur.

Farklı düzeylerde modelleme sürecinde, ilk düzeyde sistem bir bütün olarak tüm temel bileşenleri dikkate alınarak modellenir. Ardından ikinci düzeyde sistem işleyişi açısından gerekli görülen her bileşen kullanıcı hareketlerini temsil edecek şekilde bir alt model olarak yeniden modellenir. İkinci düzeydeki modelin birinci düzeydeki bileşen yerine konularak bir bütün olarak ele alındığı modeller "*detaylandırılmış / güzelleştirilmiş*" (*ing.refinement*) model olarak adlandırılır ve Şekil 2.3'de basit bir detaylandırılmış ODG örneği verilmiştir (Belli ve Budnik, 2007).

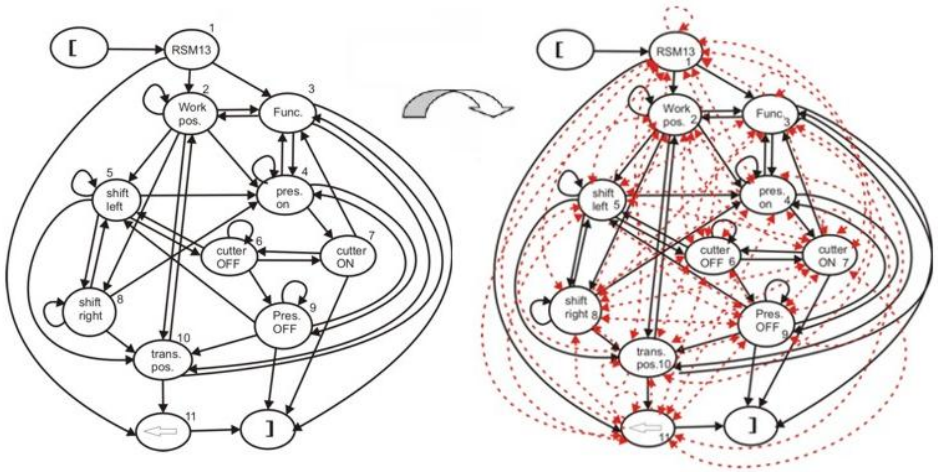
Şekil 2.3'de ODG_1 birinci düzeyde genel bir modeli temsil ederken, ODG_2, ODG_1'deki *a* olayının yada modülünün a_1, a_2 ve a_3 olaylarından oluşan alt işlem süreçlerini temsil eden modeldir. ODG_3 ise yalnızca *a* olayı açısından detaylandırılmış bir modeli temsil eder.

Bu çalışmada önerilen kümeleme ile model tabanlı test önceliklendirme metodu hem basit modeller hem de detaylandırılmış modeller için ayrı ayrı tanımlanmıştır. Basit bir model üzerinden önerilen yöntem ve tanımlamalar örneklerle açıklanacaktır.



Şekil 2.3. ODG_1, birinci düzey graf; ODG_2, ikinci düzey graf; ODG_3, detaylandırılmış graf (Budnik, 2006)

(a)



(b) **(c)**

Şekil 2.4. (a) Çim biçme makinesi ve kontrol paneli (b) Kesme biriminin beklenen hareketlerini temsil eden model (c) İstenmeyen hareketleri temsil eden sistem modeli (Budnik, 2006)

Şekil 2.4(a)'da verilen çim biçme makinesinin kontrolü için geliştirilen yazılımın kesme biriminin arzu edildiği gibi çalışması durumunu modelleyen ODG Şekil 2.4(b)'de verilmiştir. Şekil 2.4(c) ise sistemin istenmeyen davranışlarını modellemektedir. ODG'nin her bir nodu, bir olayı ve her bir arkı da olaylar arasındaki ilişkileri, geçişleri ifade eder. 11 olayla temsil edilen bu grafda olaylara ifade / gösterim kolaylığı açısından 1'den 11'e kadar numaralar verilmiştir. Bu

modele göre 1 numaralı olay gerçekleştikten sonra 2, 3 veya 11 numaralı olay gerçekleşebilir. Ancak yazılımın çalıştırılması sırasında 1 numaralı olaydan sonra Şekil 2.4(b)'de kesikli çizgilerle gösterilen geçişler gerçekleşirse örneğin 5 numaralı olay meydana gelirse bu hata olarak kabul edilir.

Örnek 2.1.1: *Model:* Bir çim biçme makinesinin "Kesme Birimini" (*ing. Cutting Unit*) test etmek için oluşturulan ODG. Daire şeklinde gösterilen ve işlem kolaylığı açısından numaralandırılmış olan tepelerin her biri sistemde gerçekleşen bir olaya karşılık gelirken oklarla gösterilen ve ark olarak adlandırılan geçişler ise sistemdeki kullanıcı hareketlerini ifade etmektedir.

2.2.Test Takımı Üretme

Yazılım test süreci test takımının üretilmesiyle başlar. Test takımı bir sistemin tam olarak gözden geçirilebilmesi için gerekli olan tüm işlem adımlarını içeren tam olay dizilerinden oluşur. Üretilen bir test takımının model üzerinde gösterilen tüm olayları ve tüm ayrıtları içermesi gerekmektedir. Ancak test sürecinde her bir işlem ek bir maliyet olduğundan test takımının uzunluğunun mümkün olduğunca küçük olması istenir. Böylece sistemdeki hataları minimum maliyetle ortaya çıkarmak amaçlanır. ÇPP olarak bilinen problemin çözümü için geliştirilen algoritmalar yardımıyla test takımları üretilebilir. Bu çalışmada test takımları ÇPP esasına dayanan Test Suite Designer Tool kullanılarak üretilmiştir.

Örnek 2.2.1: Şekil 2.4(b)'de verilen model için üretilen ve 7 tane tam olay dizisinden (TOD) oluşan bir test takımı (TT) aşağıda verilmiştir. Burada gösterim kolaylığı açısından test takımında olaylar birer sayı ile temsil edilmiştir. Sayıların hangi olaylara karşılık geldiği model üzerinde gösterilmiştir. TOD, 13 olay ve 12 arkdan oluşan bir tam olay dizisidir. 1 ,2, 3, 4, 7, 9 ve 10 olaylara karşılık gelirken, 1-3, 3-4, 4-7, …, 4-3 model üzerinde gösterilmiş arkları temsil eder.

TOD : ***[****1 3 4 7 3 10 2 2 4 4 9 4 3****]***
TT : ***{ [****1 3 4 7 3 10 2 2 4 4 9 4 3****] , [****1 2 2 5 4 7 6 6 7****], [****1 2 10 2 5 6 9 9 8 4 9 10 2 4 9****], [****1 2 4 9 5 10 11****], [****1 2 5 6 5 5 8 8 5 10 11****], [****1 2 3 2 8 10 3****], [****1 11****] }***

Bu çalışmada yalnızca sistem modeli kullanılarak test dizileri önceliklendirilmektedir. Geliştirilen önceliklendirme yöntemi girdi olarak sistem modelinden üretilmiş TOD'ları kullanmaktadır.

2.3. Olayları Nitelendiren Faktörler

Test takımını oluşturan olayların önemini ortaya çıkarabilmek için her bir olay çeşitli faktörlerle nitelendirilmiştir. Faktör sayısı kullanılan model ve sisteme göre değişkilik gösterebilir. Bu çalışmada örnek olarak 13 faktör açıklanmıştır. Tüm faktör tanımlamaları, modelin doğru kurulduğu ve test takımının optimal üretildiği varsayımı üzerinden yapılmıştır. Çalışma kapsamında başlangıçta basit bir model üzerinde gerçekleştirilen tanımlamalar daha sonra karmaşık sistemleri de kapsayacak şekilde geliştirilmiştir. Tablo 2.1'de listelendiği gibi faktör tanımlamaları üç farklı durum dikkate alınarak gerçekleştirilmiştir.

Tablo 2.1. Faktör listesi

Faktör Kapsamı	Faktör No	Faktör Açıklaması
Tek bir model	X_{i1}	Katman sayısı
	X_{i2}	Bir olayın kullanılma sıklığı
	X_{i3}	Bir olaydan ulaşılabilen olay sayısı
	X_{i4}	Alt olay sayısı
	X_{i5}	Denge derecesi
	X_{i6}	Hatalı olay çiftleri sayısı
Test dizileri	X_{i7}	Giriş olayına olan maksimum uzaklık
	X_{i8}	Test takımı içinde olayın kullanılma sayısı
	X_{i9}	Bir olayın test dizilerinde ortalama kullanılma sıklığı
Farklı düzeylerde modelleme	X_{i10}	Modelin bulunduğu düzey sayısı
	X_{i11}	Modelin sıra numarası
	X_{i12}	Bir olayın tüm düzeylerde kullanıldığı test dizisi sayısı
	X_{i13}	Giriş olayına olan minimum uzaklık

Tek bir model üzerinden 6 faktör, test dizileri dikkate alınarak 3 faktör ve hiyerarşik modellerden oluşan farklı düzeylerde modelleme durumunda 4 faktör daha eklenmesiyle toplamda 13 faktör tanımlanmıştır.

2.3.1. Tek model üzerinden tanımlanan faktörler

X_{i1}: *Katman sayısı:* bir olayın bulunduğu katman sayısı, söz konusu olayın başlangıç olayına olan minimum uzaklığını ifade eder. Katman sayısı test maliyeti açısından önemlidir, çünkü hata ortaya çıkarabilecek bir olayın başlangıç olayından uzaklığı artıkça bu hatayı düzeltmenin maliyeti artacaktır. Dolayısıyla maliyeti artırabilecek bu olayları içeren test dizilerinin ilk olarak test edilmesi gerekir.

Örnek 2.3.1.1: Şekil 2.4(b)'deki model ele alınırsa, 1 numaralı olay 1. katmanda, 2, 3 ve 11 numaralı olaylar 2. katmanda yer alırlar. Bu modeldeki olaylar 4 farklı katmanda ele alınmıştır.

X_{i2}: *Bir olayın kullanılma sıklığı*: bir olayın kullanılma sıklığı o olayın sistem içerisinde çağrılabileceği durumları ifade eder. Dolayısıyla sistem içerisinde sıklıkla kullanılan bir olayda hata meydana gelmesi sistem kalitesini olumsuz olarak etkileyeceğinden bu olayların öncelikle test edilmesi gerekir. Kullanılma sıklığı modelde tepelerle temsil edilen olayların girdi ve çıktı ayrıtlarının toplamından hesaplanır.

Örnek 2.3.1.2: Şekil 2.4(b)'deki modelde 1 numaralı olayın kullanılma sıklığı 1 girdi 3 çıktı ayrıtının toplamından 4 olarak hesaplanır. Burada 2 numaralı olay 6 çıktı 4 girdi ayrıtı ile toplamda 10 farklı durumda kullanılabilir ve dolayısıyla 2 numaralı olay 1 numaralı olaydan daha sık kullanıldığı için daha önemlidir.

X_{i3}: *Bir olaydan ulaşılabilen olay sayısı*: bir olaydan direk yada dolaylı olarak ulaşılabilecek olay sayısını ifade eder. Yazılım testi bir olaylar dizisini takip ederek gerçekleştirilir. Dolayısıyla bir olayın başarılı bir şekilde çalıştırılması kendisinden sonraki olayların işletilebilmesi için önemlidir. Bu nedenle bir olayın ulaşabileceği olay sayısı arttıkça o olayın test açısından önemi de artacaktır.

Örnek 2.3.1.3: Şekil 2.4(b)'de 1 numaralı olay doğru çalışmazsa sonraki hiçbir olayın çalıştırılması mümkün olmaz. Dolayısıyla bu faktör dikkate alındığında 1 numaralı olayın önemlilik değeri yüksektir. Bu faktör için en az öneme sahip olan olay 11 numaralı olaydır. Çünkü bu işlem başarıyla tamamladığında sadece sonuca ulaşılır. Başka hiçbir işlem için öncül bir olay değildir.

X_{i4}*: Alt olay sayısı:* yazılım sistemleri test sürecinde anlamlı olmak koşuluyla mümkün olduğunca küçük parçalara ayrılarak ve sistemi basitleştirerek modellenir. Bunun amacı hem test maliyetini düşürmek hem de çok karmaşık sistemleri bile test edebilmektir. Genel olarak bir sistemi bütün olarak test edebilmek için birden çok modeli birlikte ele almak gerekir. İşlem kolaylığı açısından parçala - fethet prensibiyle üst düzeylerde kabaca gerçekleştirilen modeller daha sonra alt graflarla modellenerek detaylandırılır ve farklı düzeylerde modelleme gerçekleştirilebilir. Dolayısıyla üst düzeylerden seçilmiş bir model bir alt düzeydeki çok sayıda modeli kapsıyor olabilir. Yani, her hangi bir model üzerinde ele alınan bir olayın bir alt düzeyde birden çok olayı kapsaması mümkündür. Bir olayın alt olaylarının sayısını dikkate alan bu faktör basit bir olayla karmaşık bir olayı ayırmayı sağlar. Alt olay sayısı fazla olan olay test açısından daha önemlidir.

Örnek 2.3.1.4: Şekil 2.4(b)'de verilen model için bu faktör dikkate alındığında en önemli olay 3 numaralı olaydır. Çünkü 10 tane alt olay içermektedir. 5, 6, 9 ve 11 numaralı olaylar yalnızca bu modelde kullanılmışlardır ve alt olay içermedikleri için 0 değerini almışlardır.

X_{i5}: *Denge derecesi (DD)*: denge derecesi bir olayın girdi ve çıktı ayrıtlarının toplamından hesaplanır. Fakat bu faktörün kullanılma sıklığından farkı girdi ayrıtları (+) işaretle, çıktı ayrıtları (-) işaretle temsil edilerek aralarında matematiksel işlem yapılmasıdır. İdeal bir olayın denge derecesinin 0 (sıfır) olması beklenir. Pozitif ve negatif değerler alabilen bu faktör için en önemli değer 0'dır. Dahası bu faktöre göre +1 ile -1 değerleri alan iki olay arasında önemlilik açısından bir farklılık yoktur.

Örnek 2.3.1.5: Şekil 2.4(b)'deki modelde 1 numaralı olayın 1 tane girdi ayrıtı (+1), ve 3 tane çıktı ayrıtı (-3) vardır. Dolayısıyla 1 numaralı olayın denge derecesi -2 olarak hesaplanır. 5 numaralı olayın +5 girdi ayrıtı ve – 5 çıktı ayrıtından dolayı denge derecesi 0 dır. Dolayısıyla bu faktör dikkate alındığında 5 numaralı olay 1 numaralı olaydan daha önemlidir.

X_{i6}: *Hatalı Olay Çiftleri (HOÇ) sayısı*: bir ODG üzerinde yer alan tüm arklar sistemin beklenen davranışlarını gösterir. Ancak sistemdeki hatalı davranışlar dahil olmak üzere meydana gelebilecek tüm olayların oluşturduğu uzay dikkate alındığında ODG deki arkların tümleyenleri hatalı olay çiftlerini gösterir. Bir olayın olası tüm davranışları ODG da gösterilmiş ise hatalı olay çifti 0 olacaktır. Hatalı olay

çifti ne kadar çoksa o olayın hata ortaya çıkarma ihtimali o kadar çok olacaktır. Bu nedenle hatalı olay çifti çok olan olayların önce test edilmesi önemlidir.

Örnek 2.3.1.1: Şekil 2.4(b)'deki modelde 1 numaralı olaydan sonra üç olay gerçekleşebilir. Bunlar 2, 3 ve 11 numaralı olaylar. Bu olayların gerçekleşmesi sistemin beklenen davranışları sergilediğini gösterir. Eğer sistem çalıştırıldığında 1 numaralı olaydan sonra 4 numaralı olay meydana geliyorsa bu hata anlamına gelmektedir. 1 numaralı olay için 3 beklenen durum varken 9 tane hatalı olay çifti mevcuttur. Bu modelde en fazla hatalı olay çifti 11 numaralı olaya aittir. Dolayısıyla bu faktör dikkate alındığında 11 numaralı olay diğerlerinden daha önemlidir.

2.3.2. Test dizileri üzerinden tanımlanan faktörler

X_{i7}: *Giriş olayına olan maksimum uzaklık*: test dizileri sıralı olaylardan meydana gelirler ve test işlemi bu sıradan yürütülür. Başarıyla işletilen bir olaydan sonra ancak bir diğer olaya geçilebilir. Her hangi bir olayda hata meydana gelirse, hata rapor edilir ve hata düzeltildikten sonra hata yakalanan test dizisi baştan tekrar çalıştırılır. Her bir test dizisini çalıştırmanın ve test dizisindeki her bir olayı yürütmenin ayrı maliyetleri vardır. Dolayısıyla hatanın bulunduğu olayın test işlemine başlangıcı temsil eden başlangıç olayından uzaklığı önemlidir. Başlangıç olayından ne kadar uzakta ise hata düzeltildikten sonraki tekrar test aşaması dikkate alındığında testin maliyeti o kadar artacaktır. Bu faktör için bir olayın başlangıç olayına olan maksimum uzaklığı ne kadar yüksekse test açısından önemide o kadar artar.

Örnek 2.3.2.1: Örnek modelden üretilen test dizileri ele alınsın

TOD_1 : [1 3 4 7 3 10 2 2 4 4 9 4 3]
TOD_2 : [1 2 2 5 4 7 6 6 7]
TOD_3 : [1 2 10 2 5 6 9 9 8 4 9 10 2 4 9]
TOD_4 : [1 2 4 9 5 10 11]
TOD_5 : [1 2 5 6 5 5 8 8 5 10 11]
TOD_6 : [1 2 3 2 8 10 3]
TOD_7 : [1 11]

Bu örnekte 1 numaralı olay tüm test dizilerinde başlangıç olayından sonraki ilk olaydır. Dolayısıyla başlangıç olayına olan uzaklığı 1 dir. 2 numaralı olay TOD_7 dışında tüm olay dizilerinde yer almaktadır ancak TOD_3 de işletilme sırası 13'dür.

Örneğin TOD_3 işletilirken 2 numaralı olaydan sonra sistem beklenmedik bir davranış sergilerse yada bir hata meydana gelirse bu hatanın düzeltilip TOD_3' ün tekrar test edilmesi, 1 numaralı olaydan sonra meydana gelen hatanın düzeltilip testin tekrarlanmasından daha pahalıya mal olacaktır. Bu faktörle tüm olaylar için maksimum uzaklık dikkate alınmıştır. Örneğin TOD_3 de 2 numaralı olay 2. ve 4. sıralarda da işletilmiştir ancak bunların hata düzeltme maliyeti daha düşük olacaktır. Bu faktör dikkate alındığında en önemli olay TOD_3 de 15. sırada işletilen 9 numaralı olaydır.

X_{i8}: *Test takımı içinde kullanılma sayısı*: bir olay bir test takımında birden çok test dizisi içinde kullanılabileceği gibi tek bir test dizisi içinde birden fazla da tekrarlanabilir. Kullanılma sayısı faktörü, söz konusu olayın tüm test dizileri içinde toplamda kaç kez işletileceğini ifade eder. Tek bir test dizi içinde bir yada birkaç kez kullanılan olaylar tüm test dizilerinde tekrarlanan olaylarla karşılaştırıldığında test açısından daha az öneme sahip olurlar.

Örnek 2.3.2.2: Örnek 2.3.2.1'de verilen test dizileri dikkate alındığında 1 numaralı olayın 7 test dizisinde birer kez olmak üzere toplamda 7 kez kullanıldığı görülmektedir. 2 numaralı olay ise TOD_6 da 3, TOD_1, TOD_3, ve TOD_4 de 2'şer, TOD_2, ve TOD_7 de 1 kez olmak üzere toplamda 11 kez çalıştırılacaktır. Dolayısıyla bu faktör dikkate alındığında 2 numaralı olay 1 numaralı olaydan daha önemlidir.

X_{i9}: *Bir olayın test dizilerinde ortalama kullanılma sıklığı*: Bir olayın kullanılma sıklığı tüm test takımında söz konusu olayın kaç kez çalıştırılacağını ifade eder. Ancak olayın kaç test dizisinde kullanıldığını ve bu test dizilerinin uzunlukları dikkate alınmaz. Bir olay için ortalama kullanılma sıklığı test dizilerinin uzunluklarını da dikkate alarak ve şu şekilde hesaplanır.

$$Avrf(x_i) = \frac{1}{d}\left(\sum_{q=1}^{r} \frac{f_q(x_i)}{l(TOD_q)}\right) \quad q = 1, \dots, r \in N; \;\; i = 1, \dots, n \in N; \;\; d \in N \qquad (2.1)$$

Burada $Avrf(x_i)$, x_i olayının ortalama kullanılma sıklığını temsil ederken; d, bu olayın kullanıldığı test sayısı; $f_q(x_i)$, q. test dizisinde x_i olayının kullanılma sıklığı; $l(TOD_q)$, q. test dizisinin uzunluğunu ifade eder.

Örnek 2.3.2.3: Örnek modelden üretilen test dizilerinin uzunlukları sırasıyla 13, 9, 15, 7, 11, 7, 2 dir. 1 numaralı olay tüm test dizilerinde 1 kez kullanılacağından ortalama kullanılma sıklığı şöyle hesaplanacaktır.

Ortalama kullanılma sıklığı = [(1/13)+(1/9)+(1/15)+(1/7)+(1/11)+(1/7)+(1/2)]/7
= 0,162

2.3.3. Farklı düzeylerdeki modeller için tanımlanan faktörler

Geniş kapsamlı yazılımların olası tüm kullanıcı davranışlarını tek bir modelle göstermek neredeyse imkânsızdır. Ayrıca model ne kadar büyürse modelin karmaşıklığı ve bu modelden üretilen test dizilerinin uzunluğu da artacaktır. Bu test açısından istenmeyen bir durumdur. Ancak modeli fazlasıyla basitleştirmek de olası sistem davranışlarını ve olası sistem hatalarının gözden kaçmasına sebep olabilir. Bu nedenle, yazılım modelleme sürecinde sistem mümkün olduğunca anlamlı küçük parçalara bölünerek modellenir. Modelleme genellik yazılım tasarım ekibi tarafından gerçekleştirilir. Ancak gerekli durumlarda testçilerde ODG ile sistemi modeleyebilir. Hiyaraşik modellemede, sistem ilk düzeyde genel olarak modellenir. Hiçbir olay detaylandırılmaz. Detaylandırma işlemi alt düzeylerde gerçekleştirilir. İlk düzeyde bir olay olarak ele alınan bir modül, ikinci düzeyde kendi içinde tekrar modellenir. Ve bu işlem yazılımın kapsamına göre dallanarak devam eder. Düzey sayısıyla ilgili bir kısıt bulunmamasına rağmen, düzey sayısı arttıkça test maliyetide üstel olarak artmaktadır. Bu nedenle gereksiz detaylandırmalardan kaçınmak gerekir.

Yukarıdaki 9 faktör tek bir model ve bu model üzerinden üretilen test dizileri için tanımlanmıştır. Birden çok model aynı anda dikkate alındığı durumlar için 4 yeni faktör daha tanımlanmıştır.

X_{i10}: *Modelin bulunduğu düzey sayısı*: farklı düzeylerde yazılım modelleme süreci basitten detaya doğru gerçekleştirildiğinden ilk düzeyde yer alan model, sistemin en genel kullanımını ifade eder. Bu modelde detay bulunmaz ve olası pek çok davranış göz ardı edilmiş olur. Bu nedenle bu düzeydeki modelin hata yakalama başarımı alt düzeylerdeki modellere göre daha düşüktür. Olaylar detaylandırılıp alt düzeylerde modellendikçe olası tüm kullanıcı davranışları denetlenebilir ve alt düzeylerdeki modellerin hataları yakalama başarımı artar. Yani modelin bulunduğu düzey sayısı sayısal değer olarak arttıkça o modelde bulunan olayların önemi de artar.

X_{i11}: *Modelin sıra numarası*: ilk düzeyde genel olarak modellenen bir graf daha sonra her olay alt düzeylerde detaylandırılarak yeniden modellenebilir. Bu durum daha alt düzeylerde de gerçekleşebilir. Aynı düzeyde birden fazla model yer aldığında bu modeller sıra numarı ile ağırlıklandırılabilir. Dolayısıyla modelin sıra numarası faktörünün değerleri söz konusu modülün önemine göre uzman görüşüne de başvurularak atanabilir.

X_{i12}: *Bir olayın tüm düzeylerde kullanıldığı test dizisi sayısı*: farklı düzeylerde modelleme durumunda test dizilerinin üretilmesi iki farklı şekilde gerçekleştirilebilir: her model için ayrı ayrı üretilerek ya da ilk düzeyden başlayarak detaylandırılmış grafları içeren bütün bir model üzerinden üretilerek. Detaylandırılmış model üzerinden üretilen test dizileri farklı düzeylerde yer alan modellerdeki olayları da içerir. Bu durumda yazılımın farklı modülleri tarafından kullanılan bir olay farklı düzeylerde ve birden çok test dizisinde yer alabilir. Dolayısıyla bir olayın tüm bir sistem içindeki kullanılma sıklığı fazla ise bu olayın önce test edilmesi gerekir. Kısaca kullanılma sıklığı değeri arttıkça olayın test açısından önemi de artacaktır.

X_{i13}: *Giriş olayına olan minimum uzaklık*: tüm düzeyler dikkate alındığında bir olayın test dizileri içinde en erken karşılaşıldığı durumu ifade eder. Söz konusu olay hata yakalama potansiyeli yüksek bir olay ise test sürecinde ne kadar erken karşılaşırsa hata o kadar erken ortaya çıkar. Ayrıca minimum uzaklık değeri arttıkça yakalanan hatanın düzeltme maliyeti de artar. Bu nedenle minimum uzaklık değeri yüksek olan olayların önemi de fazladır.

2.4. Kümeleme Tabanlı Test Önceliklendirme

Kümeleme, bir veri seti içindeki doğal grupları hakkında bilgi olmayan verilerin optimal parçalanmasını üreten tekniklerin genel ismidir. Optimal küme sayısının önceden belirlenmesi gerekir. Oluşturulan grupların kendi içinde mümkün olduğunca homojen ve grupların kendi arasında heterojen olması amaçlanır (Tatlıdil, 2002; Hoppner vd., 1999).

Bölüm 2.3'de tanımlanan faktörlerin her biri ayrı ayrı test durumlarını önceliklendirmek için test mühendislerinin yada kullanıcıların öncelikleri

doğrultusunda kullanılabilir. Ancak tek boyutlu incelemelerde olayların önemlilik dereceleri tercih edilen faktöre göre farklılık göstermektedir. Bu nedenle olayların önem derecelerini belirlemek için tüm faktörleri tek tek ele almak yerine tüm faktörlerin aynı anda dikkate alınmasına imkan sağlayan çok boyutlu veri analizi yöntemleri kullanılabilir. Böylece bir vektör olarak ele alınan olayları birbiriyle kıyaslamak ve aralarındaki farklılıkları ortaya koymak mümkün olabilir. Bu nedenle olayları önem gruplarına ayırmak amacıyla kümeleme analizi kullanılmıştır.

Bu bölümde basit model üzerinden kümeleme tabanlı test önceliklendirme metodunun açıklanmasına Şekil 2.4'de verilen örnek model üzerinden devam edilmiştir. Tek bir model ele alındığında Bölüm 2.3.1. ve Bölüm 2.3.2. de tanımlanan 9 faktör üzerinden yöntem tanıtılmıştır. Faktör sayısı metottan bağımsızdır, yani sayısının artması veya azalması metodun uygulanmasında her hangi bir değişikliğe neden olmamaktadır.

Örnek 2.5.1: Şekil 2.4'deki model için yukarıdaki tanımlanmış 9 faktörlerden oluşan ham veri seti Tablo 2.2'de verilmiştir.

Tablo 2.2. Ham veri matrisi

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	1	4	11	3	-2	9	1	7	0,162
	x_2	2	10	11	5	-2	6	13	11	0,183
	x_3	2	9	10	10	1	8	13	5	0,298
	x_4	3	10	11	2	2	8	14	8	0,174
	x_5	3	10	11	0	0	7	9	7	0,171
	x_6	4	6	11	0	-1	8	8	4	0,127
	x_7	4	5	10	3	-1	9	9	3	0,15
	x_8	4	8	11	2	0	8	9	4	0,13
	x_9	4	9	11	0	-3	6	15	6	0,162
	x_{10}	3	8	10	3	2	9	12	6	0,117
	x_{11}	2	3	1	0	1	11	11	3	0,245

Olaylar üzerinden tanımlanan tüm faktör değerleri için büyük değerler olayın önemliliği açısından daha iyi iken 5. faktör olan x_{i5} için en iyi değer 0 (sıfır)'dır. Tüm olaylarının *DD* değeri 0 olan bir graf ideal graftır ve tek bir olay dizisi ile test edilmesi mümkündür. Dolayısıyla bu değer grafın ideal graf olup olmadığını belirler. 0 dışındaki değerler grafın ideal hale getirilmesi için kullanılabilecek olayları belirlemek için kullanılır. Tablo 2.2'de verilen ham veri setinde x_{i5} faktörüne

dönüşüm uygulanarak 0 en iyi değer olacak şekilde düzenlenen yeni veri seti Tablo 2.3'de verilmiştir.

Tablo 2.3. Beşinci faktörün dönüşümü sonrası elde edilen veri seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	1	4	11	3	0,054	9	1	7	0,162
	x_2	2	10	11	5	0,054	6	13	11	0,183
	x_3	2	9	10	10	0,242	8	13	5	0,298
	x_4	3	10	11	2	0,054	8	14	8	0,174
	x_5	3	10	11	0	0,399	7	9	7	0,171
	x_6	4	6	11	0	0,242	8	8	4	0,127
	x_7	4	5	10	3	0,242	9	9	3	0,15
	x_8	4	8	11	2	0,399	8	9	4	0,13
	x_9	4	9	11	0	0,004	6	15	6	0,162
	x_{10}	3	8	10	3	0,054	9	12	6	0,117
	x_{11}	2	3	1	0	0,242	11	11	3	0,245
Ortalama(μ)		2,91	7,45	9,82	2,55	0,181	8,09	10,36	5,82	0,174
Standart Sapma(σ)		1,04	2,54	2,96	2,98	0,143	1,45	3,88	2,40	0,054

Gerekli dönüşümler yapıldıktan sonra faktörlerin dağılımları hakkında bilgi veren tanımlayıcı istatistikleri (ortalama (μ), ve standart sapma (σ)) hesaplanır. Ortalama, faktörler için merkezi değeri ifade ederken, standart sapma faktör değerlerinin yayılımı hakkında bilgi verir.

2.4.1. Gözlem ortalamalarına dayalı test önceliklendirme

Bir olayın birden çok faktörle nitelendirildiği durumlarda söz konusu olay tüm faktörlerin bileşkesinden oluşan tek bir nokta ile temsil edilebilir. Bu durumda bir olay her biri bir düzleme karşılık gelen faktörlerce temsil edilir. Ancak her faktör kendi içinde farklı bir dağılıma sahip olabilir. Bu faktörlerin ortalamaları, varyansları ve değer aralıkları da farklı olabilir. Bu durumda, olaylar arası uzaklık hesaplanırken, ortalaması veya varyansı daha büyük olan faktörler, hesaplanan uzaklık değerini daha fazla etkiler. Ayrıca faktörlerin aşırı uçlardaki değerleri kümeleme analizi sonucunda ayrı kümeler olarak da ortaya çıkabilir. Bu gibi durumlarda verilerin standardize edilmesi yada belirli aralıklarda değerlere dönüştürülmesi uygun görülmektedir (Özdamar, 2002). İstatistiksel analizlerin bu durumdan olumsuz olarak etkilenmemesi için her faktör kendi için de ortalaması ve standart sapması kullanılarak eşitlik (2.2) yardımıyla standartlaştırılmıştır.

$X \sim (\mu, \sigma^2)$ iken $\frac{X-\mu}{\sigma} = Z \sim N(\mu, \sigma^2)$ (2.2)

Tablo 2.4. Standartlaştırılmış veri seti

		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}	**Gözlem ort.**
Standartlaştırılmış gözlem değerleri	x_1	-1,83	-1,36	0,40	0,15	-0,88	0,63	-2,41	0,49	-0,23	**-0,56**
	x_2	-0,87	1,00	0,40	0,82	-0,88	-1,45	0,68	2,16	0,16	**0,23**
	x_3	-0,87	0,61	0,06	2,50	0,43	-0,06	0,68	-0,34	2,30	**0,59**
	x_4	0,09	1,00	0,40	-0,18	-0,88	-0,06	0,94	0,91	-0,01	**0,24**
	x_5	0,09	1,00	0,40	-0,86	1,52	-0,75	-0,35	0,49	-0,06	**0,16**
	x_6	1,04	-0,57	0,40	-0,86	0,43	-0,06	-0,61	-0,76	-0,89	**-0,21**
	x_7	1,04	-0,97	0,06	0,15	0,43	0,63	-0,35	-1,17	-0,46	**-0,07**
	x_8	1,04	0,21	0,40	-0,18	1,52	-0,06	-0,35	-0,76	-0,83	**0,11**
	x_9	1,04	0,61	0,40	-0,86	-1,23	-1,45	1,20	0,08	-0,23	**-0,05**
	x_{10}	0,09	0,21	0,06	0,15	-0,88	0,63	0,42	0,08	-1,07	**-0,04**
	x_{11}	-0,87	-1,75	-2,98	-0,86	0,43	2,01	0,16	-1,17	1,32	**-0,41**

Eşitlik (2.2) kullanılarak standartlaştırılmış veri seti ve bu standart değerlerden elde edilen gözlem ortalamaları Tablo 2.4'de verilmiştir.

Çok boyutlu veri uzayındaki veri noktalarını birbiriyle kıyaslamak veya sıralamak için kullanılacak en temel istatistiksel yaklaşım, gözlem ortalamalarını karşılaştırmaktır. Bu çalışmada bahsedilen olayların her biri istatistiksel anlamda birer gözlemdir. Standartlaştırma yoluyla elde edilen veri matrisinde her satır bir gözleme karşılık gelir. Her gözlem, tüm faktörlerin söz konusu gözleme ait değerlerinin ortalamasından hesaplanan ve gözlem ortalaması olarak adlandırılan tek bir değer ile temsil edilebilir. Gözlem ortalamaları ile sıralama istatistiksel olarak önerilebilecek en basit önceliklendirme yöntemidir.

Veri setinde sayısal olarak büyük olan değerler olayın hata ortaya çıkarma öneminin yüksek olduğunu gösterir. Bu nedenle olaylara önem derecelerinin atanması gözlem ortalamalarının büyükten küçüğe doğru sıralanmasıyla gerçekleştirilir. Ortalaması büyük olan olaylar test açısından daha önemlidir. Gözlem ortalamaları kullanılarak sıralanan olaylara göre test dizilerinin önceliklendirilmesi metodu gözlem ortalamalarına dayalı önceliklendirme yöntemi olarak adlandırılmış ve kümeleme tabanlı test önceliklendirmenin hata yakalama başarımını kıyaslamak için baz olarak kabul edilmiştir.

Söz konusu yaklaşımda en yüksek ortalama değerine sahip olay 1. önem sırasına sahiptir. Ortalama değeri sayısal olarak azaldıkça olayların önem dereceleride azalır. Bu şekilde olayların önem sıraları belirlendikten sonra test dizilerinin önceliğini belirlemek için kullanılacak olan önemlilik indeksi eşitlik (2.3) yardımıyla belirlenir.

Önemlilik indeksi,

$$Imp(x_i) = n - ImpD(x_i) + 1 \tag{2.3}$$

Burada $Imp(x_i)$ i. olayın önemlilik indeksi, n toplam olay sayısı ve $ImpD(x_i)$'de i. olayın önemlilik sırasını ifade etmek için kullanılmıştır.

Tablo 2.5'de görüldüğü üzere 3 numaralı olay 0,59 ortalama değeri ile bu model için en önemli olay olarak belirlenmiştir. Önemlilik sırası ve önemlilik indeksi ters orantılıdır, biri artarken diğer azalır. Örneğin, 3 numaralı olay için önemlilk sırası 1 iken önemlilik indeksi 11'dir. 1 numaralı olay ise en az önem sahip olaydır ve önemlilik sırası 11 iken önemlilik indeksi 1 olarak belirlenmiştir.

Tablo 2.5. Gözlem ortalamalarına göre olay sıraları ve önemlilik indeksleri

Olaylar	Gözlem ortalamaları	Önemlilik sırası $ImpD(x_i)$	Önemlilik indeksi $Imp(x_i)$
x_3	0,59	1	11
x_4	0,244	2	10
x_2	0,225	3	9
x_5	0,164	4	8
x_8	0,111	5	7
x_{10}	-0,035	6	6
x_9	-0,049	7	5
x_7	-0,07	8	4
x_6	-0,208	9	3
x_{11}	-0,412	10	2
x_1	-0,56	11	1

Olayların önem indeksleri belirlendikten sonra test dizilerinin öncelik değerleri eşitlik (2.4) kullanılarak hesaplanır.

$$PrefD(TOD_q) = \sum_{i=1}^{n} Imp(x_i)\, f_q(x_i) \tag{2.4}$$

$PrefD(TOD_q)$, q. test dizisinin öncelik değeri, dizide yer alan olayların önemlilik indeksleri ($Imp(x_i)$) ile frekanslarının ($f_q(x_i)$) çarpımının toplamından hesaplanır.

Öncelik değerlerinin büyükten küçüğe doğru sıralanmasıyla test dizilerinin öncelik sıralaması gerçekleştirilir. Öncelik değeri en yüksek olan test dizisi ilk olarak test edilmelidir.

Örnek 2.4.1.1: Bu örnekte x_1 olayı için önemlilik indeksinin ve TOD_1 için öncelik değerinin nasıl hesaplanacağını ele alınsın. Burada $e = 11$ ve $ImpS(x_1) = 11$ dir. Dolayısıyla, x_1 için önemlilik indeksi ($Imp(x_1)$) = 11-11+1= 1 olur.

TOD_1 : [1 3 4 7 3 10 2 2 4 4 9 4 3] için öncelik değeri ise
$PrefD(TOD_1)$= 1*1+9*2+11*3+10*4+4*1+5*1+6*1 = 107 olacaktır.

Tablo 2.6. Gözlem ortalamalarına dayalı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 7 3 10 2 2 4 4 9 4 3]	107	1
TOD_2	[1 2 2 5 4 7 6 6 7]	51	5
TOD_3	[1 2 10 2 5 6 9 9 8 4 9 10 2 4 9]	78	2
TOD_4	[1 2 4 9 5 10 11]	41	6
TOD_5	[1 2 5 6 5 5 8 8 5 10 11]	57	3
TOD_6	[1 2 3 2 8 10 3]	54	4
TOD_7	[1 11]	3	7

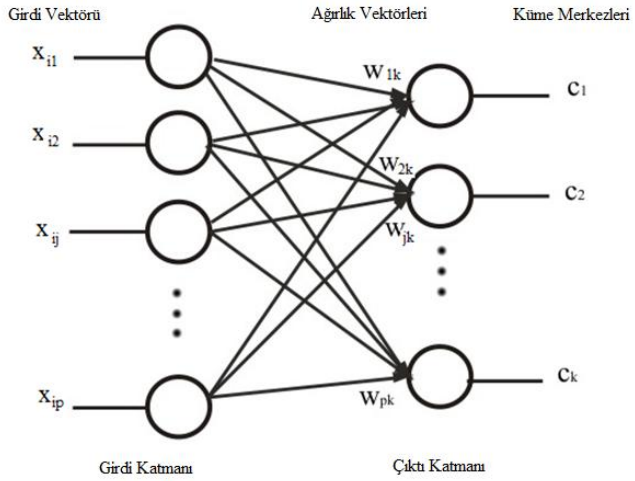
Gözlem ortalamalarına dayanan test önceliklendirme metoduyla elde edilen sıralama Tablo 2.6'da verilmiştir. Buna göre TOD_1 birinci önceliğe sahip test dizisidir. Sonraki sıralamalar ise TOD_3, TOD_5, TOD_6, TOD_2, TOD_4, TOD_7 şeklinde olacaktır. Bu sıralama aynı modelden kümeleme yöntemleriyle elde edilen sıralamalar için baz alınacaktır.

2.4.2. Klasik kümeleme – yapay sinir ağlar

Yapay sinir ağları gözlemlerin veya faktörlerin dağılımlarıyla ilgili varsayımlara ihtiyaç duymamaları nedeniyle istatistiksel yöntemler yerine pek çok uygulamada kullanılmaktadır. Ayrıca istatistiksel bir kümeleme yöntemi olan k-ortalamalar ile çok boyutlu ölçekleme yöntemlerinin her ikisinin de işlevini yerine getirebilmesi nedeniyle kümeleme çalışmalarında Kohonen ağları tercih edilmektedir (Fu,1994).

Kohonen ağları girdi ve çıktı katmanı olmak üzere iki katmandan oluşur. Girdi katmanında giriş nöronları bulunur ve giriş nöronlarının sayısını veri setindeki faktör sayısı (p) belirler. Çıktı katmanı aynı zamanda yarışmacı katman olarak da

adlandırılır. Çıkış nöronlarının her biri bir kümeyi (*c*) temsil eder. Girdi katmanı ile çıktı katmanındaki nöronların her biri, bir biri ile bağlantılıdır ve ağ ağırlık vektörü olarak adlandırılan bu bağlantıların değerlerinin güncellenmesi ile ağ eğitilir. Yapay sinir ağlarının esnekliği ve hata tolere edebilme özelliği, bilginin ağ üzerinde ağırlık vektörleri aracılığıyla dağınık olarak tutulmasından kaynaklanır. Yarışmacı öğrenme algoritması ile eğitilen Kohonen ağları, yarışmacı öğrenme ağları olarak adlandırılır. Ağ yapısı Şekil 2.5'de verilmiştir.



Şekil 2.5. Yarışmacı öğrenme ağ yapısı (Fu, 1994)

Yarışmacı öğrenme algoritması "kazanan her şeyi alır" prensibiyle çalışan danışmansız bir öğrenme algoritmasıdır. Söz konusu eğitme prensibine göre her adımda rasgele seçilen tek bir girdi vektörü ile tüm ağırlık vektörleri arasındaki uzaklık belirlenir ve yalnızca girdi vektörüne en yakın olan ağırlık vektörü eğitilir. Amaç girdi vektörlerini benzerliklerine göre gruplamak ve küme merkezlerine karşılık gelen ağırlık vektörlerini belirlemektir. Eğitim işlemi ağırlık vektörlerinin değerlerindeki değişim yeterince küçük belli bir sayıya yada deneme yanılma yoluyla karar verilen bir iterasyon sayısına ulaşıncaya kadar devam eder. Ağın eğitimi tamamlandıktan sonra tüm girdi vektörlerinin kendilerine en yakın küme merkezinin temsil ettiği kümeye atanmasıyla sınıflandırma işlemi yapılır. Sınıflandırma işlemi klasik kümeleme mantığına dayanır yani her eleman yalnız bir kümenin üyesi olabilir. Her elemanın dahil olduğu kümeye aitlik derecesi de 1'dir.

Ağırlık vektörlerinin başlangıç değerleri veri seti içersinden rasgele seçilir. Özellikle geniş veri setlerinde (geniş veri setleri n>30 olarak alınabilir) kümeleme sonuçları

ağırlık vektörlerinin başlangıç değerlerinden olumsuz etkilenebilir. Başlangıç değerleri rasgele seçildiğinde veri seti içerisinde aykırı değerler bulunuyorsa bu değerler başlangıç değeri olarak seçilebilir ve bunun sonucunda aykırı kümeler oluşabilir. Aykırı kümeler tek bir elemandan da oluşabilir. Bu bölünmeli kümeleme algoritmalarının genel bir sorunudur. Bu nedenle daha iyi bir kümeleme performansını elde edebilmek için kümeleme işlemini makül bir sayıda tekrarlamak ve içlerinden kümeleme hatası küçük olanı seçmek tercih edilen bir yoldur. Bir diğer yaklaşımda uyarlamalı yarışmacı öğrenme algoritması olarak bilinen küme merkezlerinin elenmesine dayalı metottur. Buna göre başlangıçta çok sayıda küme merkezi belirlenerek ağırlık vektörleri eğitilir ve her eğitme işleminden sonra istenen küme sayısına ulaşana kadar küme içi hatası küçük olan küme merkezleri birer birer silinerek istenen küme sayısına ulaşana kadar eğitme işlemi tekrarlanır. Böylece ağırlık vektörlerinin başlangıç değerlerine olan bağımlılık yok edilmiş yada azaltılmış olur. Uyarlamalı yarışmacı öğrenme algoritması geniş veri setlerinde daha iyi sonuçlar vermektedir. Ancak eleme mekanizmasından dolayı işlem maliyeti artacağından küçük veri setlerinde ağırlık vektörlerinde uyarlamaya gerek kalmadan standart yarışmacı öğrenme ile eğitim yeterli görülmektedir.

2.4.3. Yarışmacı öğrenme

Test önceliklerini belirmek amacıyla kullanılacak olan yarışmacı öğrenme (YÖ) algoritması kendi kendini düzenlemeli ağ modellerinin bir türüdür (Kohonen, 1989). YÖ algoritmasına göre "*kazanan*" ağırlık vektörü (girdi ve çıktı birimleri arasındaki bağlantıların ağırlıkları) veya küme merkezleri üzerinde çalışılan YSA'nın eğitim aşamasında *"kazanan her şeyi alır (ing. winner-takes-all)"* stratejisi uygulanarak ayarlanabilir (Haykin, 1994; Fu, 1994). En basit anlamıyla ağın eğitim işlemi veri uzayındaki girdi vektörlerini en iyi temsil edecek küme merkezlerini belirleme sürecidir. Her adımda rasgele seçilen bir girdi vektörüne en yakın ağırlık vektörü güncellenir. Yakınlık ölçüsü olarak ise öklit uzaklığı veya açı ölçüleri kullanılır.

YÖ algoritmaları $X = \{x_1, \dots, x_i, \dots, x_n\} \subset \mathbb{R}^p$ girdi uzayında her hangi bir $x_i = \{x_{i1}, \dots, x_{ip}\} \in \mathbb{R}^p$ girdi vektörüne göre kazanan ağırlık vektörünün $w_k = \{w_{k1}, \dots w_{kc}\} \in \mathbb{R}^p$ belirlenmesi kullanılan benzerlik ölçütüne göre uzaklığa dayalı

YÖ algoritması ve açıya dayalı YÖ algoritması olmak üzere ikiye ayrılır. Uzaklığa dayalı YÖ algoritmasında girdi vektörü ile ağırlık vektörlerinin benzerliği öklit uzaklığına göre belirlenir. Açıya dayalı YÖ algoritması benzerlik ölçüsü olarak birim vektöre dönüştürülmüş bir hiper uzayda girdi vektörleri arasındaki açıyı kullanır.

2.4.3.1. Uzaklığa dayalı yarışmacı öğrenme

Yarışmacı öğrenme ağları ağırlık vektörlerinin bir biriyle yarıştı ve yalnızca kazanan ağırlık vektörünün değerinin güncellendiği ağlardır. Kazanan ağırlık vektörü, girdi vektörüne en yakın ağırlık vektörüdür. Benzerlik ölçüsü olarak öklit uzaklığının kullanıldığı uzaklığa dayalı YÖ algorimasına göre *k*. kazanan ağırlık vektörü, $w_k \in \mathbb{R}^p$, eşitlik (2.5) kullanılarak belirlenir [Fu, 1994; Maeda, vd., 1996]:

$$w_k = \arg\min_k \left\{ \left\| x_i - w_k \right\| \right\} \qquad i = 1,2,...,n \qquad k = 1,2,...,c \tag{2.5}$$

$$\Delta w_k = \eta (x_i - w_k) \tag{2.6}$$

burada η öğrenme oranıdır. Öğrenme oranı ağırlık vektörünün güncellenmesindeki kullanılan adım büyüklüğünü belirleyen bir sabittir. Bu değerin büyük olması ağırlık vektörünün değişimi sırasında büyük sıçramalara neden olur, küçük olması ise eğitim süresini uzatır. Kazanan ağırlık vektörü güncelleme değeri, Δw_k, eşitlik (2.6) ile hesaplanır. Güncelleme değeri ağırlık vektörünün girdi vektörüne yaklaşma adımıdır. Ve bu değer kazanan ağırlık vektörünün eski değerine eklenerek ağırlık vektürünün yeni değeri, w_{k_yeni} eşitlik (2.7) yardımıyla belirlenir.

$$w_{k\_yeni} = w_{k\_eski} + \Delta w_k \tag{2.7}$$

2.4.3.2. Açıya dayalı yarışmacı öğrenme

Benzerlik ölçüsü olarak birim vektörler arasındaki açı ölçüsü kullanılan açıya dayalı yaklaşımda hem veri noktaları $\tilde{x}_i$ hem de $\tilde{w}_k$ ağırlık vektörleri birim uzunlukta olacak şekilde normalleştirilir (Rumelhart ve Zipser, 1985). Birim vektör, uzunluğu 1 olan vektördür.

Bu yaklaşımda kazanan ağırlık vektörü, $\tilde{w}_w$, eşitlik (2.8)'de görüldüğü gibi birim uzunluktaki girdi vektörleri $\tilde{x}_i$, ile birim uzunluktaki $\tilde{w}_k$ ağırlık vektörleri arasındaki en küçük açıya sahip vektör olarak belirlenir.

$$\tilde{\mathrm{w}}_{\mathrm{w}} = \arg\max_{k} \left\{ \sum_{j=1}^{p} \tilde{x}_{ij} \tilde{w}_{kj} \right\} \quad i = 1, ..., n \quad k = 1, ..., c \tag{2.8}$$

Ya da $\cos\theta$'nın maksimum değeri 1 olduğundan söz konusu vektörler arasındaki açı dikkate alınarak eşitlik (2.9) ile belirlenir.

$$\tilde{\mathrm{w}}_{\mathrm{w}}^{\theta} = \arg\min_{k} \left\{ \theta_k \right\} \quad k = 1, ..., c \in R \tag{2.9}$$

Kazanan ağırlık vektörü $\Delta\tilde{w}_w$ için güncelleme kuralı eşitlik (2.10)'da verilmiştir. (Rumelhart, Zipser, 1985).

$$\Delta\tilde{w}_w = \eta(\frac{\tilde{x}_i}{p} - \tilde{w}_w) \tag{2.10}$$

Kazanan ağırlık vektörünün güncellenmiş değeri, $\tilde{\mathrm{w}}_{\mathrm{w\_yeni}}$, güncelleme değeri, $\Delta\tilde{\mathrm{w}}_{\mathrm{w}}$, eklenerek eşitlik (2.11)'de verildiği gibi hesaplanır.

$$\tilde{w}_{w\_yeni} = \tilde{w}_{w\_eski} + \Delta\tilde{w}_w \tag{2.11}$$

Yarışmacı öğrenme ağlarının eğitimi küme merkezlerine karşılık gelen ağırlık vektörlerinin değerlerinin ayarlanması sürecidir. Eğitim süresi ise önceden belirlenen sonlandırma kriterlerine göre değişiklik gösterir. İki farklı sonlandırma kriteri kullanılabilir. Ağın eğitimi ya önceden belirlenen bir iterasyon sayısına ulaşıncaya kadar yada güncelleme değeri olan $\Delta\tilde{\mathrm{w}}_{\mathrm{w}}$'nin değeri yeterince küçük bir sayıya ulaşıncaya kadar devam eder.

Yeterince büyük veri setleri ağırlık vektörlerinin başlangıç değerlerinin rasgele seçilmesinden olumsuz etkilenirler. Bu durumda önceden belirlenen küme sayısından çok daha fazla ağırlık vektörü ile ağ eğitilmeye başlanır. Her eğitme işlemi sonucunda kümeleme yoluyla veriler sınıflandırılır ve eşitlik (2.12) ile hesaplanan sınıflandırma hatası belirlenir (Martinez, vd., 1993; Maeda ve Miyajima, 2000).

$$D_k = \frac{1}{p}\left(\sum_{\tilde{x} \in S_k} \tilde{x}\tilde{w}_w\right) \quad k = 1, \ldots, c \tag{2.12}$$

Burada D_k, k. kümenin sınıflandırma hatası, p, faktör sayısı, $\tilde{x} \in S_k$ k. kümenin elemanları ve $\tilde{w}_w$ kazanan ağırlık vektörüdür. D_s, minimum sınıflandırma hatasına sahip kümeyi temsil etmek üzere $D_k \geq D_S$ koşulunu sağlayan ağırlık vektörünün temsil ettiği küme merkezi silinerek yeni küme sayısı ile eğitim işlemi tekrarlanır (Maeda, vd., 1996; Maeda ve Miyajim, 2000; Eminov ve Gökçe, 2005). Eğitim işlemi optimal küme sayısına ulaşıncaya kadar devam eder.

En iyi kümeleme performansını elde etmek için, ağırlık vektörlerinin optimal küme sayısına ulaşıncaya kadar silinmesine dayanan uyarlamalı yarışmacı öğrenme (UYÖ) algoritması Tablo 2.7'de verilmiştir.

Tablo 2.7. Uyarlamalı yarışmacı öğrenme (UYÖ) algoritması

Adım 1. *Başlangıç*

Eleman sayısı *n*, başlangıç küme sayısı *k(0)=m*, optimum küme sayısı *L*, öğrenme oranı η, Ağırlık vektörlerinin başlangıç değerleri $\{w_1(0), \ldots, w_k(0), \ldots, w_l(0)\}$, döngü sayısı t ve maksimum döngü sayısını T_{max}, ata.

Adım 2. *Standart Yarışmacı Öğrenme:*

2.1. *X* veri kümesinden rasgele bir $\tilde{x}_i$ girdi vektörü seç.

2.2. Eşitlik (2.8)'i kullanarak $\tilde{w}_w$ kazanan ağırlık vektörünü belirle

2.3. Eşitlik (2.10)'u kullanarak kazanan $\tilde{w}_w$ vektörünü güncelle.

2.4. Döngü değerini bir arttır. $t \leftarrow t + 1$

2.5. Sonlandırma kriteri sağlanmış ise Adım 3'e git, değilse adım 2.1'e dön.

Adım 3. *Eleme Mekanizması:*

3.1. Eşitlik (2.12)'i kullanarak sınıflandırma hatalarını hesapla

3.2. $D_k \geq D_S$ kontrol et ve küçük D_S hesaplanan $\tilde{w}_s$'i ele.

3.3. Başlangıç küme sayısını bir azalt $m \leftarrow m - 1$.

Adım 4. *Sonlandırma Koşulu*:

$t = T_{\max}$ ise sonlandır, değilse Adım 2'e git.

Sınıflandırma: Küme merkezlerine karşılık gelen ağırlık vektörlerinin değerleri belirlendikten sonra, sırasıyla veri setindeki her eleman

$$S_k = \left\{ x \in R^p \,\middle|\, \sum_{j=1}^{p} \tilde{x}_{ij} \tilde{w}_{kj} \geq \sum_{j=1}^{p} \tilde{x}_{ij} \tilde{w}_{mj} \;\; \forall k \neq m \right\} \tag{2.13}$$

koşulunu sağlayacak biçimde c tane kümeden kendisine en yakın olan kümeye atanarak klasik mantığa göre sınıflandırılır. Optimal küme sayısının c'nin önceden belirlenmesi gerekir. Bu çalışmada c'nin belirlenmesinde küme içi ve kümeler arası uzaklıkları dikkate alması nedeniyle V_{sv} algoritması kullanılmıştır (Kim, vd. 2001; Eminov, 2003).

2.4.4. Yapay sinir ağ tabanlı test önceliklendirme

Bu çalışmada test dizilerini sıralamak amacıyla yarışmacı öğrenme algoritması ile kümelemeye dayalı bir test önceliklendirme metodu önerilmiştir. Sistem modeli ve model üzerinden üretilen test dizileri kullanılarak yukarıda tanımlanmış faktörlerden elde edilen çok boyutlu veri seti YÖ algoritması ile kümelenir. Olayların YÖ algoritması kullanılarak kümelenmesi sonucu elde edilen kümeler kendi aralarında ortalama vektörlerinin uzunlukları kıyaslanarak bir önem sırasına göre sıralanır. Uzunluğu en büyük olan küme merkezinin temsil ettiği olaylar test açısından en önemli olaylar olacaktır. Olaylara önem dereceleri atanması yoluyla eşitlik (2.3) ve (2.4) kullanılarak test dizileri önceliklendirilir.

Tablo 2.8. Yarışmacı öğrenme ile kümeleme sonuçları

Kümeler $S_k^{(YÖ)}$	**Olaylar**	**Üyelik değeri**	**Ortalama vektör uzunluğu $l(\bar{x}_k)$**	***Önemlilik sırası $ImpD(S_k)$***	***Önemlilik indeksi $Imp(S_k)$***
$S_1^{(YÖ)}$	x_4	1	21,711	2	3
	x_5	1			
	x_9	1			
	x_{10}	1			
$S_2^{(YÖ)}$	x_1	1	17,294	3	2
	x_7	1			
	x_8	1			
$S_3^{(YÖ)}$	x_6	1	16,065	4	1
	511	1			
$S_4^{(YÖ)}$	x_2	1	23,299	1	4
	x_3	1			

Örnek 2.4.4.1: Yine şekil 2.4(b)'de verilen model örnek olarak ele alınsın. Bu model için elde edilen ve Tablo 2.3'de verilmiş olan veri seti yarışmacı öğrenme algoritması ile kümelenmiş ve sonuçlar Tablo 2.8'de verilmiştir.

Tablo 2.8'de $S_k^{(YÖ)}$, yarışmacı öğrenme algoritması sonucu elde edilen kümeleri temsil eder. Bu algoritma klasik kümeleme algoritması olduğu için tüm elemanların kümeye aitlik durumları yani üyelik değerleri 1 dir. Kümeleme sonrası küme merkezlerine karşılık gelen küme ortalama vektörlerinin uzunlukları $\left(l(\bar{x}_k)\right)$ hesaplanır ve bu değerler büyükten küçüğe sıralanarak kümelerin önemlilik sıraları belirlenir. 2 ve 3 numaralı olaylardan oluşan $S_4^{(YÖ)}$ kümesinin ortalama vektörünün uzunluğu 23,299 ile en fazladır. Dolayısıyla bu olaylar söz konusu model için en önemli olaylardır. Fakat üyelik değerleri 1 olmasından dolayı bu olaylar arasında hem önemlilik hemde olayların kümeyi temsil dereceleri açısından bir fark yoktur.

Test önceliklendirme sürecinde olayların atandıkları kümelerin önemlilik indeksleri kullanılarak testlerin öncelik dereceleri belirlenir. Tablo 2.9'da verilen sonuçların nasıl elde edildiği Örnek 2.4.4.2'de anlatılmıştır.

Tablo 2.9. Yarışmacı öğrenmeye dayalı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 7 3 10 2 2 4 4 9 4 3]	42	2
TOD_2	[1 2 2 5 4 7 6 6 7]	22	5
TOD_3	[1 2 10 2 5 6 9 9 8 4 9 10 2 4 9]	44	1
TOD_4	[1 2 4 9 5 10 11]	19	6
TOD_5	[1 2 5 6 5 5 8 8 5 10 11]	27	3
TOD_6	[1 2 3 2 8 10 3]	23	4
TOD_7	[1 11]	3	7

Örnek 2.4.4.2: TOD_3 dizisinin önemlilik derecesinin yarışmacı öğrenme algoritmasıyla kümeleme sonucunun nasıl belirlendiği ele alınsın. Bu dizide yer alan 1 numaralı olay 3. önemlilik derecesine sahip kümenin elemanıdır ve önemlilik indeksi 2'dir. 2 ve 3 numaralı olaylar ise en önemli olaylardır. Sırasıyla dizideki tüm elamanların önemlilik indeksleri toplanarak Örnek 2.4.1.1'deki gibi TOD_3 dizisinin öncelik değeri şu şekilde hesaplanır:

TOD_3 : [1 2 10 2 5 6 9 9 8 4 9 10 2 4 9] için

$PrefD(TOD_3)$=1*2+3*4+2*3+1*3+1*1+4*3+2*3= 44

Kümelemeye dayalı test önceliklendirme metodunun mevcut yaklaşımlardan farkı test dizilerinin uzunluklarından çok dizileri oluşturan olayların önemini ortaya çıkarmasıdır.

2.4.5. Bulanık kümeleme – bulanık c-ortalamalar

Klasik kümeleme yaklaşımı {0,1} mantığına dayanır ve kesinlik içerir. Yani bir birim bir kümenin ya elemanıdır yada değildir. Oysa bulanık mantıkta bir birim aynı anda birden fazla kümenin elemanı olabilir ve [0, 1] aralığında bütün değerleri alabilir (Zadeh, 1965). Bulanık kümeleme algoritmalarından en popüler olanı Dunn tarafından önerilen ve Bezdek tarafından geliştirilen bulanık c-ortalamalar (BCO) (*ing. Fuzzy c-means, FCM*) algoritmasıdır (Dunn, 1973; Bezdek, 1981). BCO algoritması, verilerin sıfır ile bir arasında farklı üyelik değerleriyle (*ing: membership degrees*) bir veya daha fazla kümeye ait olabilme mantığına dayanan bulanık kümeleme algoritmasıdır. Bir birim aynı anda birden çok kümenin elemanı olabilir ancak en yüksek üyelik değeri ile ait olduğu kümeye atanarak sınıflandırılır. Bulanık kümeleme klasik kümelemeye göre daha esnektir. Bu nedenle gruplar arası sınırların kesin olmadığı durumlarda bulanık kümeleme daha iyi sonuçlar üretmektedir.

Bulanık c-ortalamalar algoritması en küçük kareler yönteminin genellemesi olan amaç fonksiyonunun minimizasyonuna dayanır.

$$J(X,U,V) = \sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ki})^m d^2\,(v_k, x_i) \qquad (2.14)$$

Burada *X* veri matrisi, *U* üyelik değerleri, *V* küme merkezlerini temsil eder. Bulanık küme sayısı *c*, *i*. olayın *k.* kümeye aitlik derecesini gösteren üyelik değerleri $u_{ki} \in [0,1]$, *k.* küme merkezi $v_k \in \mathbb{R}^p$, *k.* küme merkezi ile *i.* olay arasındaki öklit uzaklığı $d(v_k, x_i)$ ve bulanıklık indeksi *m* ile temsil edilir. *m* bulanıklık indeksi değiştirilerek kümeleme sonuçları değiştirilebilir. Ancak birçok paket program *m=2* olarak almaktadır. Bu çalışmada BCO algoritması ile kümeleme için MATLAB'ın bulanık toolbox'ın da bulunan hazır fonksiyon kullanılmıştır ve *m* değeri 2 alınmıştır. BCO algoritması küme merkezlerinin başlangıç değerleri rasgele seçer. Dolayısıyla YÖ algoritmasında olduğu gibi BCO algoritması da küme merkezlerinin başlangıç

değerlerine bağımlıdır. Bu nedenle en iyi sonucu elde etmek için birden çok deneme yapmak gerekebilir.

Veri noktalarının birbirinden ayrık kümeler oluşturduğu bir uzayda klasik ve bulanık kümeleme sonuçları benzerlik gösterir. Ancak veri yoğunluğunun sık olduğu ya da örtüşen kümelerden oluşan veri uzaylarında klasik kümeleme küme sınırlarını belirlemekte yetersiz kalırken bulanık kümelemenin daha iyi sonuçlar verdiği bilinmektedir.

2.4.6. Bulanık c-ortalamalara dayalı test önceliklendirme

Bu yaklaşımda veri setinin kümelenmesinde BCO algoritması kullanılmıştır. Bulanık kümeleme sonucunda elde edilen kümeler kendi aralarına önem derecelerine göre sıralanmıştır. Kümeleme sonuçları ile eşitlik (2.3) kullanılarak belirlenen önemlilik sıraları ve önemlilik indeksleri Tablo 2.10'da verilmiştir.

Tablo 2.10. Bulanık c-ortalamalar ile kümeleme sonuçları

Kümeler $S_k^{(BCO)}$	**Olaylar**	**Üyelik değeri**	**Ortalama vektör uzunluğu** $l(\bar{x}_k)$	***Önemlilik sırası*** $ImpD(S_k)$	***Önemlilik indeksi*** $Imp(S_k)$
$S_1^{(BCO)}$	x_5	0,4941	18,566	2	3
	x_6	0,8064			
	x_7	0,7477			
	x_8	0,9548			
$S_2^{(BCO)}$	x_2	0,7363	22,518	1	4
	x_3	0,4523			
	x_4	0,9294			
	x_9	0,6414			
	x_{10}	0,5217			
$S_3^{(BCO)}$	x_{11}	0,997	16,282	4	1
$S_4^{(BCO)}$	x_1	0,9949	16,674	3	2

Tablo 2.10'da $S_k^{(BCO)}$, bulanık c-ortalamalar algoritması sonucu elde edilen kümeleri temsil eder. Bulanık kümeleme algoritmasına göre her elemanın kümeye aitlik durumunu, üyelik değerleri belirler. Bu üyelik değerleri 0 ile 1 arasındaki tüm değerleri alabilirler. Bu örnek için kümeleme sonucunda ortaya çıkan üyelik değerleri tabloda verilmiştir. Bulanık kümeler oluşturulduktan sonra küme merkezlerine karşılık gelen ortalama vektörlerinin uzunlukları $\left(l(\bar{x}_k)\right)$ hesaplanır ve

bu değerler büyükten küçüğe sıralanarak kümelerin önemlilik sıraları belirlenir. Tablo 2.10'da verilen sonuçlardan görüldüğü gibi 1 ve 11 numaralı olaylar tek başlarına birer küme olarak ayrılmıştır. Bu noktaların diğerlerinden farklı olduğu anlamına gelir. Bunlar aykırı olaylar olarak da düşünülebilir. Bu çalışmada olayların tek tek incelenmesinin sonuçlar açısından bir sakıncası olmadığından ayrık kümeler göz ardı edilmemiştir. Küme ortalama vektörlerini uzunluğu incelendiğinde $S_2^{(BCO)}$ kümesinin ortalama vektörünün uzunluğu 22,518 ile en fazladır. Dolayısıyla bu kümeye ait olan olaylar ele alınan model için en önemli olaylardır. $S_2^{(BCO)}$ kümesi ele alınırsa bu kümeye dahil olan olaylar arasında 4 numaralı olayın 0,9294 ile kümeyi en iyi temsil eden olay olduğu ve sıralamanın 4, 2, 9, 10 ce 3 numaralı olay şeklinde olduğu görülür. Bu küme içerisinde 4 ile 3 numaralı olayın kümeyi temsil etme dereceleri arasında yaklaşık %47 oranında bir farklılık vardır. Bu farklılık küme içinde olayları sıralama imkanı da sağlar. Dolayısıyla test sıralamaları üzerinde de etkili olacaktır.

Bulanık c-ortalamalar ile test dizilerinin öncelik değerleri

$$PrefD(TOD_q) = \sum_{i=1}^{n} Imp(x_i)\,\mu_{S_k}(x_i) f_q(x_i) \;\; q = 1, \dots, r \in N \qquad (2.15)$$

kullanılarak hesaplanır.

$PrefD(TOD_q)$, q. test dizisinin öncelik derecesi, dizide yer alan olayların önemlilik indeksleri ($Imp(x_i)$), i. olayın k. kümeye aitlik derecesini veren üyelik değeri $\mu_{S_k}(x_i)$ ile frekanslarının ($f_q(x_i)$) çarpımlarının toplamından hesaplanır.

Örnek 2.5.5.1: TOD_1 dizisinin önemlilik derecesinin bulanık c-ortalamalar algoritmasıyla kümeleme sonucu nasıl belirlendiği ele alınsın. Burada bir önceki örnekten farklı olarak üyelik değerleri de dikkate alınacaktır. Bu dizide yer alan 1 numaralı olay üyelik değeri 0,9949 ile 2. önemlilik derecesine sahip kümenin elemanıdır. Test dizilerinin öncelik sırası diziyi oluşturan olayların önemlilik indeksleri ile üyelik değerlerinin çarpımlarının toplamından hesaplanır.

TOD_1 : [1 3 4 7 3 10 2 2 4 4 9 4 3] için öncelik değeri

$PrefD(TOD_1)$ = 2*0,9949*1 + 4*0,4523*3 + 4*0,9294*4 + 3*0,7477*1 + 4*0,5217*1
+ 4* 0,7363*2 + 4* 0,6414*1
= 35,0737

olarak hesaplanır. Tüm test dizileri için elde edilen sonuçlar Tablo 2.11'de verilmiştir.

Tablo 2.11. Bulanık c-ortalamalara dayalı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 7 3 10 2 2 4 4 9 4 3]	35,0737	2
TOD_2	[1 2 2 5 4 7 6 6 7]	22,4047	3
TOD_3	[1 2 10 2 5 6 9 9 8 4 9 10 2 4 9]	39,4625	1
TOD_4	[1 2 4 9 5 10 11]	15,7843	6
TOD_5	[1 2 5 6 5 5 8 8 5 10 11]	22,0960	4
TOD_6	[1 2 3 2 8 10 3]	16,4498	5
TOD_7	[1 11]	2,9868	7

2.5. Sonuçlar

Bu çalışmada önerilen kümeleme ile model tabanlı test önceliklendirme yöntemi başlığında ele alınan üç farklı yaklaşımla yapılan sıralama sonuçları Tablo 2.12'de verilmiştir. Baz yöntem olarak gözlem ortalamaları ile önceliklendirme kullanılmış ve iki farklı kümeleme yaklaşımına göre elde edilen sıralamalar istatistiksel olarak karşılaştırılmıştır.

Elde edilen sıralamalar arasında anlamlı bir farklılık olup olmadığı Friedman testi ile test edilmiştir ve sonuçlar Tablo 2.13'de verilmiştir. İlişkili örneklemleri karşılaştırmak amacıyla kullanılan Friedman testine göre (anlamlılık değeri < 0,05 ise metotlar arasında anlamlı bir farklılık vardır; anlamlılık değeri > 0,05 ise metotlar arasında anlamlı bir farklılık yoktur) 0,028 anlamlılık değeri ile kullanılan önceliklendirme metoduna göre elde edilen sıralamalar arasında anlamlı bir farklılık vardır. Bu sonuç test dizilerinin sıralamaların tesadüfi olamayacak kadar birbirinden farklı olduğunu göstermektedir.

Tablo 2.12. Karşılaştırmalı sonuçlar

No	Test Dizisi	Öncelik Sıraları		
		GO	***YÖ***	***BCO***
TOD_1	[1 3 4 7 3 10 2 2 4 4 9 4 3]	1	2	2
TOD_2	[1 2 2 5 4 7 6 6 7]	5	5	3
TOD_3	[1 2 10 2 5 6 9 9 8 4 9 10 2 4 9]	2	1	1
TOD_4	[1 2 4 9 5 10 11]	6	6	6
TOD_5	[1 2 5 6 5 5 8 8 5 10 11]	3	3	4
TOD_6	[1 2 3 2 8 10 3]	4	4	5
TOD_7	[1 11]	7	7	7

Tablo 2.13. Friedman testi sonuçları

	Test İstatistikleri
Ki-Kare	7,143
Serbestlik Derecesi	2
Anlamlılık Değeri	0,028

Üç farklı yaklaşımla elde edilen sıralamaların ikili karşılaştırılması ve sıralamalar arasında anlamlı bir ilişki olup olmadığı Kendall Tau testi ile incelenmiş ve test sonuçları Tablo 2.14'de verilmiştir.

Korelasyon katsayısının anlamlı olması elde edilen sıralamaların tesadüfi olmadığının istatistiksel göstergesidir. Korelasyon katsayısı -1 ve +1 arasında değerler alabilir. Katsayı değerinin 1'e yakın olması, söz konusu iki sıralama arasında güçlü bir ilişki olduğu anlamına gelir. 0'a yakın olması zayıf ilişkiyi gösterir. Korelasyon katsayısının işareti negatif olduğunda negatif yönlü bir ilişkiyi, pozitif olduğunda doğrusal bir ilişkiyi ifade eder. Korelasyon değerinin anlamlılığı belirli anlamlılık düzeylerinde test edilebilir. %5 anlamlılık düzeyinde Tablo 2.14'de verilmiş olan 0,78 korelasyon katsayısı değeri anlamlıdır. Ayrıca YÖ ve GO ile elde edilen sıralamalar arasında istatistiksel olarak doğrusal, pozitif yünlü ve kuvvetli bir ilişki olduğu anlamına gelir. Yine Tablo 2.14'de YÖile BCO yada GO ile BCO için anlamlılık değerleri sırasıyla 0,224 ve 0,176 olduğundan söz konusu yöntemler arasındaki ilişkiyi ifade eden korelasyon katsayısı %5 anlamlılık seviyesinde anlamlı değildir.

Tablo 2.14 Kendall Tau testi sonuçları

			YÖ	GO	BCO
Kendall's Tau_b	YÖ	Korelasyon katsayısı	1,000	0,781	0,390
		Anlamlılık (çift-taraflı)	0,000	0,015	0,224
	GO	Korelasyon katsayısı		1,000	0,429
		Anlamlılık (çift-taraflı)		0,000	0,176
	BCO	Korelasyon katsayısı			1,000
		Anlamlılık (çift-taraflı)			0,000

2.5.1. Maliyetlerin karşılaştırılması

Önceliklendirme yöntemlerinin başarımlarını ölçmek için hata bilgisinin bulunmadığı durumlarda literatürde kapsam kriteri yada maliyet kriteri kullanılmaktadır. Bu bölümde kullanılan modellerin oluşturulduğu sistem için hata

bilgisine ulaşmak mümkün olmadığından önerilen yöntemler kapsam kriterlerine göre karşılaştırılmıştır.

Önceki bölümde test önceliklendirme süreci bir model üzerinde örnek olması açısından ayrıntılı bir şekilde anlatılmıştır. Benzer şekilde önerilen test önceliklendirme yöntemi 3 ayrı modele daha uygulanmış ve kullanılan diğer modeller ile çalışmaların detayları Ek 2.1'de verilmiştir. Tablo 2.15'de verilen sonuçlarda, 4 model için test maliyetleri birinci öncelikli test dizileri dikkate alınarak ark sayısı ve node sayısı kriterine göre karşılaştırılmıştır. Kapsam kriterleri 4 başlıkla incelenmiştir. Bu kriterler ilk önceliğe sahip test dizilerinin test takımındaki toplam olay ve ark sayısının ne kadarını kapsadığını ifade etmektedirler. Ayrıca maliyet açısından her test dizisinin kapsadığı olay sayısı o test dizisinin maliyetine de karşılık gelir. Çünkü test bütçesi incelenecek test dizisi saysı ve test dizilerinin uzunlukları dikkate alınarak yönetilir.

Test önceliklendirme yöntemlerinin amacı testin maliyetini düşük tutmaya çalışırken sistemdeki hataları da mümkün olduğunca erken ortaya çıkarmaktır. Kapsam kriteri ele alınan sistemin mümkün olduğunca çok kısmını incelemeye yönelikken maliyet kriteri kısa test dizilerini seçme eğilimindir. Tablo 2.15'de her üç metoda göre birinci önceliğe sahip test dizileri kapsam kriterine göre karşılaştırılmış ve gözlem ortalamaları kullanılarak yapılan sıralama maliyetinin kümeleme metotları ile yapılan sıralamadan daha düşük bir maliyete sahip olduğu görülmüştür.

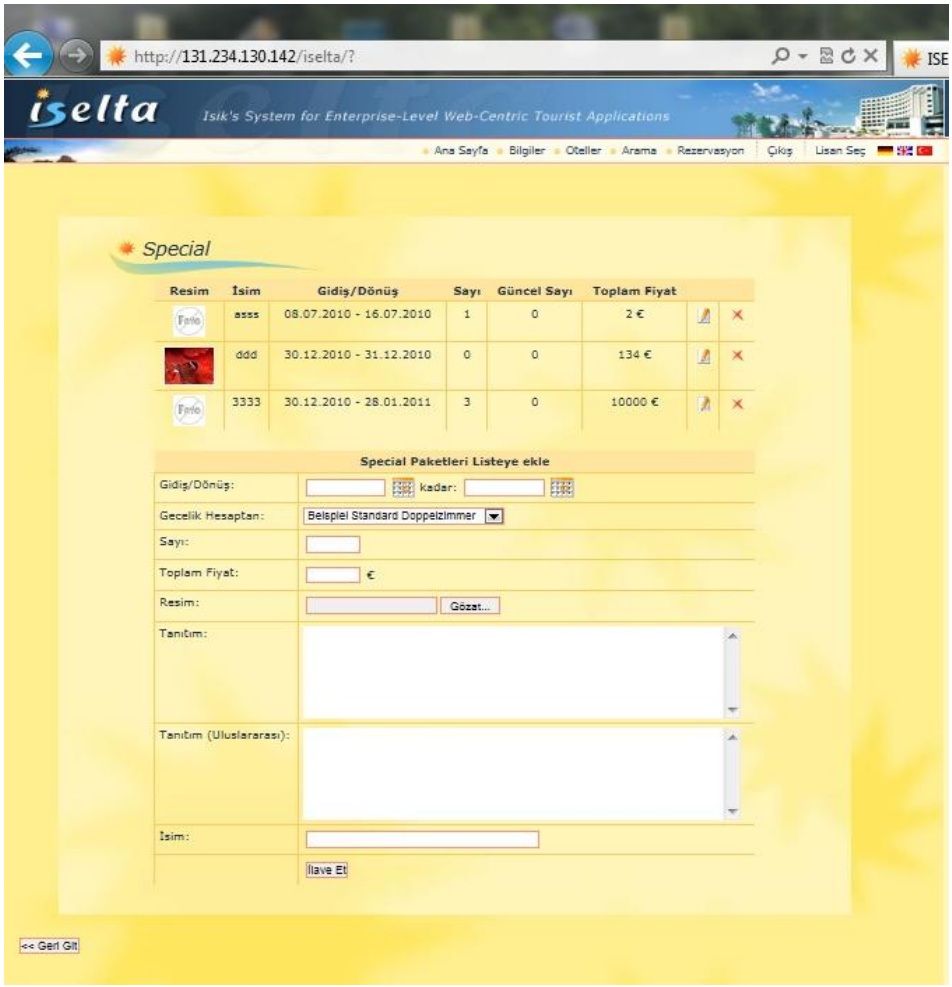
Bu bölümde incelenen modellerin oluşturulduğu yazılımların hata bilgisi olmadığından test önceliklendirme yönteminin hataları erken ortaya çıkarmadaki başarımı hakkında bir şey söylenemez. Bu nedenle önerilen yöntem online rezervasyon sistemi olan ISELTA üzerine uygulanmış ve test sonucunda ortaya çıkarılan hatalar kaydedilerek, önceliklendirme yönteminin kullanılan metotlara göre hata bulma başarımları son bölümde tartışılmıştır.

Tablo 2.15. Maliyet açısından karşılaştırmalı sonuçlar

		Ark Kapsamı		Nod Kapsamı	
		Tekrarsız	**Tekrarlı**	**Tekrarsız**	**Tekrarlı**
GO	ODG_1	0,3077	0,2143	0,6364	0,2031
	ODG_2	0,3529	0,24	0,7	0,2258
	ODG3	0,4090	0,2195	0,5333	0,1778
	ODG4	0,3878	0,3390	0,8182	0,3043
Ortalama	**Maliyet**	**0,3644**	**0,2532**	**0,6718**	**0,2278**
YÖ	ODG1	0,3333	0,25	0,7273	0,2344
	ODG2	0,5882	0,4	0,7	0,3548
	ODG3	0,3636	0,1951	0,4667	0,2
	ODG4	0,5102	0,4237	0,9091	0,3913
Ortalama	**Maliyet**	**0,4489**	**0,3172**	**0,7008**	**0,2951**
BCO	ODG1	0,3333	0,25	0,7273	0,2344
	ODG2	0,5882	0,4	0,7	0,3548
	ODG3	0,3636	0,1951	0,4667	0,2
	ODG4	0,5102	0,4237	0,9091	0,3913
Ortalama	**Maliyet**	**0,4489**	**0,3172**	**0,7008**	**0,2951**

3. BULGULAR VE İRDELEME

Geliştirilen önceliklendirme yöntemi bir turizm uygulaması olan ISELTA (Isık's System for Enterprise-Level Web-Centric Tourist Applications) yazılımının test sürecinde kullanılmıştır. ISELTA, oteller ve turizm acentaları tarafından online rezervasyonlar yapılmasına yönelik bir sistemdir. Orta ölçekli bir turizm acentası olan Işık Turistik Ltd. ile Paderborn Üniversitesinin işbirliği ile geliştirilmektedir (URL).



Şekil 3.1. ISELTA'nın özel indirimler / fırsatlar modülünün ekran görüntüsü

Tez çalışmasında önerilen önceliklendirme yöntemi ISELTA'nın nispeten küçük bir parçası olan ve Şekil 3.1'de bir ekran görüntüsü verilen "Özel İndirimler / Fırsatlar Modülüne (*ing. Special Module*)" uygulanmıştır. Bu modül kullanılarak her hangi bir otel için özel indirimler, özel oda tipleri ve ya özel rezervasyon tarihleri tanımlanabilir. Bunun yanı sıra önceden tanımlanmış olan özel fiyat veya rezervasyon türleri değiştirilip kaldırılabilir.

Bu çalışmada test öncesi TestSuiteDesigner Tool kullanılarak ODG ile söz konusu modülün modellemesi yapılmıştır. Oluşturulan modeller üzerinden test dizilerinin üretilmesinde iki farklı yol izlenmiştir: her model için test takımı üretilmesi, modül için tek bir test takımının üretilmesi. Kümeleme ile test önceliklendirme yöntemi her iki yol içinde ayrı ayrı uygulanmıştır. İlk uygulamada söz konusu modül 3 düzeyde 5 ODG kullanılarak modellenmiş ve Bölüm 3.1'de her model için test dizilerinin ayrı ayrı üretilmesi durumunda test önceliklendirmek sürecinin nasıl işleyeceği uygun stratejiler verilerek açıklanmıştır. Bölüm 3.2'de verilen ikinci uygulamada ise ilk uygulamada kullanılan modeller biraz daha geliştirilerek ve yine 3 düzeyde ancak toplamda 6 ODG üzerinden söz konusu sistem modellenmiştir. İkinci uygulamada test dizileri ilk uygulamadan farklı olarak bütün sistemi kapsayan detaylandırılmış model üzerinde üretilmiştir. Bu noktada söz konusu test takımını oluşturan test dizilerinin farklı düzeylerdeki modellerde yer alan olayları da kapsadığına dikkat etmek gerekir.

3.1. Uygulama I

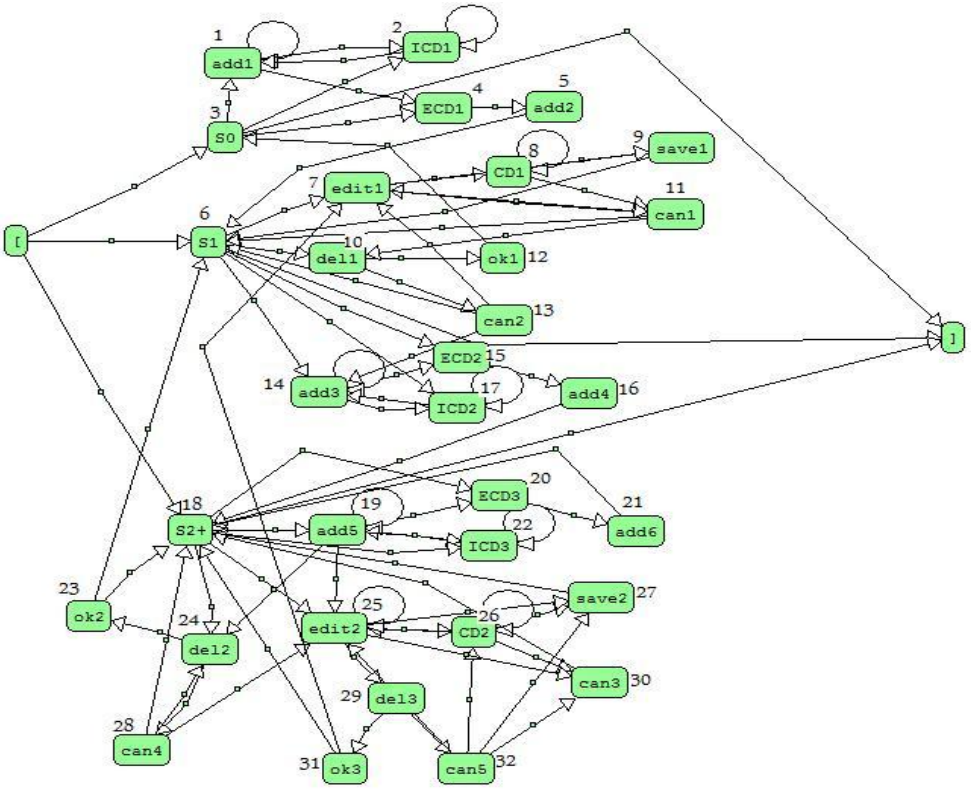
Bu bölümde tez çalışması süresince geliştirilen kümeleme tabanlı test önceliklendirme yöntemi farklı düzeylerde modellenmiş bir sisteme genişletilerek özel indirimler modülünün testine uygulanmıştır. Bu uygulamada söz konusu modül 3 farklı düzeyde 5 ayrı ODG ile modellenmiştir. Şekil 3.1 de verilen ODG özel indirim modülünün birinci düzeydeki temel grafı olan IS'i (*ing. **I**SELTA **S**pecial*) temsil eder. Bu modelde S0, S1 ve S2+ (ayrıca sırasıyla 1, 3 ve 6 ile gösterilen) olayları sistemde işlem yapılan anda bulunan tanımlı özel fırsat sayılarını ifade etmek için kullanılmışlardır. S0 (*ing. Zero Special*) herhangi bir özel indirimin bulunmadığını ifade eder. Yani her hangi bir kayıt mevcut değildir. S1 (*ing. only one*

special) yalnızca bir özel indirim ve S2+ (*ing. two or more specials*) ise iki veya daha fazla sayıda tanımlanmış özel indirim olduğunu ifade eder.

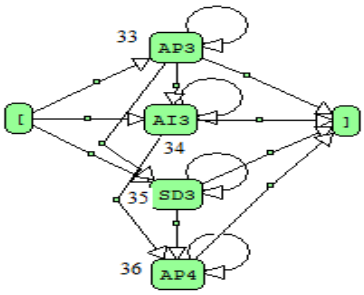
IS'de görülen ICD (*ing. Enter **I**n**C**omplete **D**ata*) olayı eksik veri girişinin gerçekleştiği olayları göstermek için kullanılmıştır. İfade kolaylığı olması açısından modelde gösterilen olaylar 1'den 32'ye kadar sayılarla temsil edilmiş ve Tablo 3.2 ile Tablo 3.4'de olaylara karşılık bu sayılar kullanılmıştır.

ICD olayı, 2 (ICD1), 17 (ICD2) ve 22 (ICD3) numaralı olaylarla modülün çalışması sürecinde kullanılma durumuna göre üç farklı şekliyle eksik veri girişi olaylarını temsil etmiştir. ECD (ing. **E**nter **C**omplete **D**ata) olayı tam veri girişi yapılması durumunu ifade etmek için kullanılmıştır. ECD olayı 4 (ECD1), 15 (ECD2) ve 20 (ECD3) numaralı olaylarla özel indirim tanımlarken kullanıcıların tam veri girişi gerçekleştirebileceği üç farklı durumdaki olayları temsil eder. Benzer şekilde CD (*ing. **C**hange **D**ata*) olayı da mevcut veriyi değiştirme olayı olarak modelde 8 (CD1) ve 26 (CD2) numaralı olaylar ile temsil edilmiştir. İlk düzeydeki modelde bir olay olarak gösterilen ICD, ECD ve CD gerçek sistemde birden çok olayın gerçekleşmesi durumunu ifade etmektedir. Örneğin, veri girişi olayı özel indirim tanımlama bilgilerinin girilmesi, fotoğraf eklenmesi tarih seçilmesi gibi birden çok alt olayı içermektedir. Veri girişi için zorunlu tüm alanların doldurulması durumu ECD ile bir kısmının yada tamamının boş bırakılması durumları da ICD olaylarına karşılık gelmektedir.

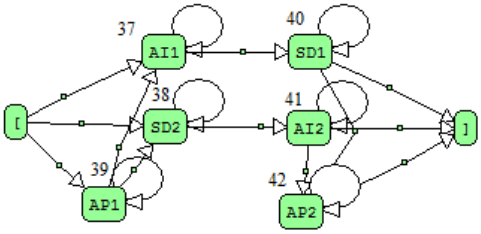
Şekil 3.2'de verilen ICD grafı yeni bir indirim türü tanımlama sürecinde eksik veri girişi olayının gerçekleşeceği kullanıcı hareketlerini temsil eden modeldir. Benzer şekilde ECD ve CD olaylarıda sırasıyla Şekil 3.3 ve Şekil 3.4 de tam veri girişi yapılması ve veri değiştirme sürecinde kullanıcı hareketerini temsil eden alt olayları gösterecek şekilde modellenmişlerdir. İkinci düzeyde yer alan bu modellerde 35 (SD3), 38 (SD2), 40 (SD1) ve 49 (SD) numaralı olaylarla temsil edilen tarih seçme olayı, SD (*ing. **S**elect **D**ate*), özel indirimin tanımlanacağı tarih aralığını seçerken olası tüm kullanıcı hareketlerini ifade edecek şekilde yeniden modellenmiş ve üçüncü düzey bir graf olarak Şekil 3.5'de verilmiştir.



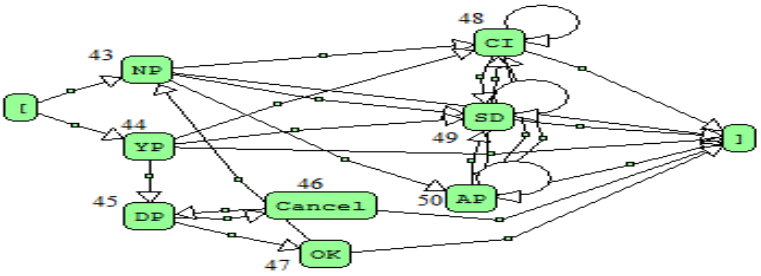
Şekil 3.2. *IS*: Özel indirim modülü birinci düzeydeki temel ODG modeli



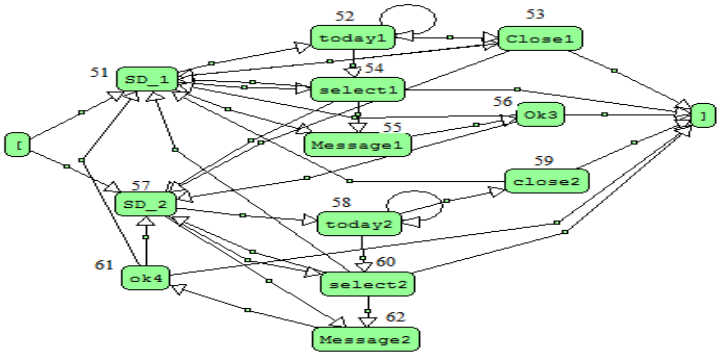
Şekil 3.3. *ICD:* Eksik veri girişini temsil eden ODG modeli



Şekil 3.4. *ECD:* Tam veri girişini temsil eden ODG modeli



Şekil 3.5. *CD*: Veri değiştirme olaylarını temsil eden ODG modeli



Şekil 3.6. *SD:* Tarih seçme adımlarını temsil eden ODG modeli

Tablo 3.1. Modelde yer alan olayların açıklamaları

S0: Özel indirim yok	DP: Fotoğraf sil
S1: Tek özel indirim	CI: Bilgi değiştir
S2+: İki veya daha fazla özel indirim	SD: Tarih seç
ICD: Eksik veri girişi	AP: Fotoğraf ekle
ECD: Tam veri girişi	Cancel: İptal butonu
CD: Veri değiştirme	OK: Tamam butonu
add: Ekleme butonu	AI: Giriş ekle
edit : Değişiklik butonu	SD_1: Varış tarihi seç
ok: Tamam butonu	SD_2: Ayrılma tarihi seç
save: Kaydet butonu	today: Bugün butonuna tıkla
can: İptal butonu	close: Kapat butonuna tıkla
NP: Fotoğraf yok	select: Tarih Seç
YP: Fotoğraf var	Message: Hata mesajı

Modülün modellenmesinde kullanılan olayların açıklamaları Tablo 3.1'de verilmiştir. Söz konusu modül için optimal test takımları *TestSuiteDesigner Tool* ile üç düzeyde

modellenen her model için ayrı ayrı üretilmiş ve elde edilen test takımları Tablo 3.2'de listelenmiştir.

Tablo 3.2 Modellerden üretilen test dizilerinin listesi

Düzeyler	ODG	No	TOD
Birinci Düzey	IS	TOD_1	[3 1 1 4 5 6 7 8 8 9 6 7 8 11 6 10 12 3 4 5 6 10 13 6 10 13 7 9 6 14 14 15 16 18 19 19 20 21 18 19 25 25 26 25 26 26 27 18 20 21 18 25 27 18 25 30 18 25 29 31 18 25 29 31 7 11 7 11 10 13 14 15 16 18 25 29 32 25 29 32 26 30 18 25 29 32 27 18 24 28 18 24 28 25 29 32 30 18 24 28 24 23 6 15 16 18]
		TOD_2	[18 22 22 19 22 19 24 23 18]
		TOD_3	[3 2 2 1 2 1 4 5 6]
		TOD_4	[6 17 17 14 17 14 15 16 18]
		TOD_5	[3]
İkinci Düzey	ICD (2, 17, 22)	TOD_6	[33 33 34 34 36 36]
		TOD_7	[33 35 35 36]
		TOD_8	[35]
		TOD_9	[34]
		TOD_{10}	[33]
	ECD (4, 15, 20)	TOD_{11}	[37 37 40 40 42 42]
		TOD_{12}	[38 38 41 41 42]
		TOD_{13}	[39 39 38 41]
		TOD_{14}	[39 37 40]
	CD (8, 26)	TOD_{15}	[43 48 48 49 48 50 48]
		TOD_{16}	[44 45 46 45 46]
		TOD_{17}	[43 49 49 50 49]
		TOD_{18}	[44 45 47 43]
		TOD_{19}	[44 45 47]
		TOD_{20}	[43 50 50]
		TOD_{21}	[44 49]
		TOD_{22}	[44 48]
		TOD_{23}	[44]
Üçüncü Düzey	SD (35, 38, 40, 49)	TOD_{24}	[51 52 52 54 51 52 53 51 54 57 58 58 59 51 54 55 56 51 54]
		TOD_{25}	[57 58 60 51 53 57 60 57 60 62 61 51 53]
		TOD_{26}	[51 55 56 57 58 59]
		TOD_{27}	[57 62 61 57 62 61]
		TOD_{28}	[51 55 56]
		TOD_{29}	[57 60]

3.1.1. Test Maliyeti

Farklı modellerden üretilen test takımlarının test sürecinde kullanılması durumunda, alt düzeylerdeki test dizilerinin olası tüm permütasyonlarının bir üst modelde yerine yerleştirilerek çalıştırılması gerekir. Bu ise testin maliyetini üstel olarak arttırır. Çoğu zaman tüm olası durumları denemek için proje bütçesi yeterli olmaz. Bu uygulamada

tüm olası durumları test etmenin toplam test maliyeti eşitlik (3.1) kullanılarak belirlenmiştir (Belli ve Budnik, 2006).

$$\psi_{TOD} = \beta.\#TOD + \alpha.l_{kap} \quad (3.1)$$

Burada proje bütçesine göre önceden belirlenen β, bir test dizisini test için başlangıç maliyeti, ve α, her bir işlem adımını gösteren tıklama maliyetine karşılık gelir. $\#TOD$, test dizilerinin toplam sayısı ve l_{kap} toplam olay sayısını veya test dizilerinin toplam uzunluğunu temsil eder. Bu uygulamada, ele alınan modülün testi için gerekli toplam maliyetler farklı düzeylere göre hesaplanmış ve sonuçlar Tablo 3.3'de verilmiştir. Tablodaki değerlerden açıkça görüldüğü gibi detaylandırılma düzeyi arttıkça incelenmesi gereken test dizisi sayısı da üstel olarak artmaktadır.

Tablo 3.3. Her düzey için belirlenen test maliyetleri

Detaylandırma Düzeyi	ψ_{TOD}	**Ortalama l_{kap}**	$\#TOD$	**Test dizisi sayısı**
Birinci Düzey (Detaylandırma yok)	$5\,\beta + 134\,\alpha$	27	5	59
Birinci + İkinci Düzey	$8.10^9\,\beta + 1{,}5.10^{13}\,\alpha$	198	$\sim 8.10^9$	656
Birinci + İkinci + Üçüncü Düzeyler	$10^{30}\,\beta + 1{,}6.10^{33}\,\alpha$	1424	$\sim 10^{30}$	5368

İkinci düzeydeki modellerle detaylandırılmış bir modelde test maliyeti, alt modeller için üretilen test dizilerinin tüm permütasyonlarının ilk düzeyde temsil ettikleri olay yerine yerleştirilmesiyle hesaplanır. Aynı şekilde üçüncü düzeydeki modellerle detaylandırılmış modelin test maliyeti önce üçüncü düzeydeki test dizilerinin tüm permütasyonlarının ikinci düzeydeki modele yerleştirilmesi, sonrasında ise ikinci düzeydeki test dizilerinin tüm permütasyonlarının ilk düzeydeki modele yerleştirilmesiyle hesaplanır. Tabloda verilen maliyetler yaklışık değerlerdir.

3.1.2. Kümeleme tabanlı test önceliklendirme

Oluşturulan modeller üzerinde üretilen test dizilerinin sıralaması önerilen test önceliklendirme yöntemi ile gerçekleştirilmiş ve sonuçlar Tablo 3.4'de verilmiştir. Tüm modeller için birinci önceliğe sahip test dizileri ortaktır. IS modelinde üretilen test dizileri arasında BCO ile sıralama sonucu TOD_3 ve TOD_4 2. ve 3. öncelik

sırasına sahipken, UYÖ ile sıralama sonucunda TOD_3 ve TOD_4 aynı öncelik değeri ile 2. önceliğe sahiptir.

Tablo 3.4. Test dizilerinin kümeleme ile önceliklendirme sonuçları

Düzeyler	ODG	No	TOD	Öncelik Derecesi	
				UYÖ	BCO
Birinci Düzey	IS	TOD_1	[3 1 1 4 5 6 7 8 8 9 6 7 8 11 6 10 12 3 4 5 6 10 13 6 10 13 7 9 6 14 14 15 16 18 19 19 20 21 18 19 25 25 26 25 26 26 27 18 20 21 18 25 27 18 25 30 18 25 29 31 18 25 29 31 7 11 7 11 10 13 14 15 16 18 25 29 32 25 29 32 26 30 18 25 29 32 27 18 24 28 18 24 28 25 29 32 30 18 24 28 24 23 6 15 16 18]	332	224,48
		TOD_2	[18 22 22 19 22 19 24 23 18]	30	20,11
		TOD_3	[3 2 2 1 2 1 4 5 6]	31	28,15
		TOD_4	[6 17 17 14 17 14 15 16 18]	31	23
		TOD_5	[3]	3	3,14
İkinci Düzey	ICD (2, 17, 22)	TOD_6	[33 33 34 34 36 36]	10	5,68
		TOD_7	[33 35 35 36]	6	3,47
		TOD_8	[35]	1	0,79
		TOD_9	[34]	1	0,95
		TOD_{10}	[33]	1	0,94
	ECD (4, 15, 20)	TOD_{11}	[37 37 40 40 42 42]	20	5,41
		TOD_{12}	[38 38 41 41 42]	17	4,45
		TOD_{13}	[39 39 38 41]	11	3,67
		TOD_{14}	[39 37 40]	9	2,71
	CD (8, 26)	TOD_{15}	[43 48 48 49 48 50 48]	22	6,45
		TOD_{16}	[44 45 46 45 46]	14	4,88
		TOD_{17}	[43 49 49 50 49]	18	3,83
		TOD_{18}	[44 45 47 43]	11	3,86
		TOD_{19}	[44 45 47]	8	2,90
		TOD_{20}	[43 50 50]	9	2,92
		TOD_{21}	[44 49]	6	1,58
		TOD_{22}	[44 48]	5	1,92
		TOD_{23}	[44]	2	0,95
Üçüncü Düzey	SD (35, 38, 40, 49)	TOD_{24}	[51 52 52 54 51 52 53 51 54 57 58 58 59 51 54 55 56 51 54]	55	15,94
		TOD_{25}	[57 58 60 51 53 57 60 57 60 62 61 51 53]	41	11,56
		TOD_{26}	[51 55 56 57 58 59]	21	5,08
		TOD_{27}	[57 62 61 57 62 61]	20	5,41
		TOD_{28}	[51 55 56]	10	2,40
		TOD_{29}	[57 60]	7	1,83

Detaylandırılmış modellerle test maliyetindeki artış dikkate alındığında olası tüm durumları incelemenin güçlüğü ortaya çıkmaktadır. Bu nedenle yine tüm test dizilerini çalıştırma imkanı olmadığı göz önüne alınarak farklı bir test sıralama stratejisi geliştirilmiştir.

3.1.3. Farklı düzeylerde test önceliklendirme stratejileri

Bu bölümde kümeleme ile model tabanlı test önceliklendirme yönteminin farklı düzeylerdeki birden çok model üzerinden ayrı ayrı üretilen test takımlarının sıralanması yoluyla sistem testini gerçekleştirmeye yönelik stratejiler 3 aşamada açıklanmıştır.

Strateji 1: İlk düzeyde, test dizileri öncelik derecelerine göre sıralanır ve testler bu sıra ile çalıştırılır. Bu örnekte, TOD_1 iki kümeleme sonucuna göre de ilk test edilmesi gereken test dizisi olarak belirlenmiştir.

Strateji 2: Bu aşamada ikinci düzeydeki test dizileri öncelik derecelerine göre büyükten küçüğe doğru sıralanır ve ilk düzeydeki test dizilerinde ikinci düzeyde detaylandırılarak yeniden modellenmiş olan olaylardan üretilen test takımından öncelik değerleri yüksek test dizileri önce test sürecine katılır.

Örnek 3.1.3.1: Birinci ve ikinci düzey testleri şu şekilde çalıştırılır: TOD_1 de bulunan 4, 8, 15, 20 ve 26 numaralı olaylar detaylandırılmış olaylardır. Bu nedenle, test işlemi sırasında detaylandırılmış bu olayların çalıştırılması gerektiğinde bu olaylar için gelişitirilen modelden üretilen ve ikinci düzeyde gerçekleştirilen test sıralamasında en yüksek önceliğe sahip testler TOD_{11}, TOD_{15}, TOD_{11}, TOD_{11}, TOD_{15} sırasıyla işletilir.

Strateji 3: Bu aşamada, ikinci ve üçüncü düzeydeki detaylandırılmış olaylar yerine test dizileri içinde en yüksek öncelik derecesine sahip test dizileri ilk olarak test edilecek şekilde ilk düzey testleri çalıştırılır.

Örnek 3.1.3.2: Birinci, ikinci ve üçüncü düzey testleri şu şekilde çalıştırılır: TOD_1 de bulunan 4, 8, 15, 20 ve 26 numaralı olaylar detaylandırılmış olaylardır. Bu nedenle, test sürecinde ikinci düzeyde yapılan test sıralamasında en yüksek önceliğe sahip testler TOD_{11}, TOD_{15}, TOD_{11}, TOD_{11}, TOD_{15} sırasıyla işletilir. Bunun yanında, TOD_{11} de 40 ve TOD_{15} de 49 numaralı olaylar da üçüncü düzeyde detaylandırılmış olaylardır ve üçüncü düzey test dizilerinin öncelik sıralaması en yüksek olan test dizileri ile yer değiştirilerek çalıştırılırlar.

3.1.4. Sonuçlar

Bu çalışmada farklı düzeylerde modellenmiş yazılımlara yönelik test önceliklendirme stratejileri ile üretilen test dizilerinin olası tüm permütasyonlarını çalıştırmak yerine önceliği yüksek testlerin ilk olarak test edilmesi önerilmiştir. Olası tüm durumları denemek ya da rasgele bir seçim yapmak yerine testleri bir öncelik sıralamasına göre işletmek önemli bir avantajdır. Bu stratejilere dayanarak gerçekleştirilen test sonucunda özel indirim modülünde 12 hata ile karşılaşılmıştır. Hataların düzeylere göre ortaya çıkış sayıları Tablo 3.5'de verilmiştir.

Tablo 3.5. Düzeylere göre karşılaşılan hata sayısı

Detaylandırma Düzeyi	***Karşılaşılan Hata Sayısı***	***Ek Hata***
Birinci Düzey (Detaylandırma yok)	8	8
Birinci + İkinci Düzey	11	3
Birinci + İkinci + Üçüncü Düzeyler	12	1

Tablo 3.6 Düzeylere göre karşılaşan hataların listesi

Düzeyler	**TOD**	**Karşılaşılan Hatalar**
Birinci Düzey	TOD_1	0 adet özel indirim tanımlaması
		Çalışmayan Today butonu
		Geçmiş tarihli başlangıç zamanı kaydetme.
		Geçmiş tarihli bitiş zamanı kaydetme.
		Başlangıç tarihinin otomatik düzeltilmemesi
		Bitiş tarihinin otomatik olarak düzeltilmemesi
		Eksik hatalı dosya uyarısı
	TOD_2	Eksik tarihle özel indirim tanımlama
İkinci Düzey	TOD_1	Başlangıç tarihi geçmişte kalmış ise ayrılış tarihini geçerli tarihli olarak seçme
		Bir özel indirim için tanılanan sayı ve fiyatlar hatalı olduğunda uyarı eksik.
	TOD_2	Boş tarihler için hata mesajı eksik
Üçüncü Düzey	TOD_2	Başlangıç ve bitiş tarihleri için geçmiş tarihli tanımlama

Tablo 3.6'da ise test sonucunda karşılaşılan hataların listesi verilmiştir. TOD_1 detaylandırma yapmaksızın çalıştırıldığında 7 hata ile karşılaşılmıştır. Ardından TOD_2 nin çalıştırılması sırasında ilk hatalardan farklı olarak 1 yeni hata ortaya çıkmıştır. İkinci düzeyde modellerin ilk öncelikli test dizilerinin birinci düzey modelde temsil ettiği olay yerine yerleştirilmesi sonucu gerçekleştirilen test sonucu TOD_1'de 2, TOD_2'de ise 1 yeni hata ortaya çıkmıştır. Aynı şekilde üçüncü düzeydeki

modelin ilk öncelikli test dizisinin üst modellerde yenine koyulması sonucu ortaya çıkarılan 1 yeni hata ile toplam 12 hata yakalanmıştır. Bu uygulamada elde edilen sonuçlar düzey sayısı artarken, test maliyetinin arttığını ancak ortaya çıkarılan yeni hata sayısının azaldığını göstermiştir. Düzey sayısı arttıkça bir hatayı yakalama maliyetide artmaktadır.

Bu uygulamada ele alınan modüldeki hata sayısı önceden bilinmediği için yalnızca önceliklendirme stratejilerine göre ortaya çıkarılan hata sayısı verilmiştir. Test dizilerinin tüm olası kombinasyonlarının çalıştırılması sonucu başka hatalarda ortaya çıkarılabilir. Hatasız bir yazılım olamayacağı gerçeği unutulmamalıdır.

3.2. Uygulama II

İkinci uygulama da ISELTA'nın özel indirimler modülü ilk uygulamaya göre nispeten geliştirilerek yine 3 düzeyde ancak toplamda 6 ODG'dan oluşacak şekilde modellenmiştir. Bu uygulamada modelleri oluşturmak ve modellerden test dizileri üretmek için *TestSuiteDesigner Tool* kullanılmıştır [TSD-Uygulamalı Bilişim, Paderborn Üniversitesi] (Belli vd., 2010). Ayrıca her model için ayrı ayrı test takımı üretmek yerine tüm ikinci ve üçüncü düzeydeki modellerle detaylandırılmış ilk düzey graf üzerinden tek bir test takımı üretilmiştir. Bu test takımındaki test dizileri, farklı düzeylerdeki modellerde yer alan olayları da kapsamaktadır. Bu yolla test dizilerinin uzunlukları artmış ancak test dizisi sayısının ve test maliyetinin aşırı miktarda artması engellenmiştir.

3.2.1. Farklı düzeylerde modelleme

Bölüm 3.1'deki yapıya benzer olarak örnek modül 3 farklı düzeyde 6 grafdan oluşacak şekilde modellenmiştir. Modelleme ve önceliklendirme metotlarının modele uygulanma stratejileri Bölüm 3.1'de ayrıntılı bir şekilde açıklandığı için bu bölümde fazla detaya girilmeden elde edilen sonuçlara odaklanılmıştır. Ancak özel indirimler modülünün detaylandırılmış ODG modelleri Ek3.1'de, bu modeller üzerinden üretilen test takımı da Ek 3.2'de verilmiştir. Sistem bir bütün olarak ele alındığında 95 olay ve bu olayların oluşturduğu 89 test dizisi incelenerek sıralanmıştır. Test

önceliklendirme metotlarının başarımlarını hata bulma açısından karşılaştırabilmek amacıyla sisteme 50 tane hata eklenmiş ve bu hatalar Ek 3.3'de verilmiştir.

3.2.2. Test önceliklendirme

Önerilen test önceliklendirme yaklaşımlarının test dizilerine nasıl uygulandığı önceki bölümlerde ayrıntılı bir şekilde açıklandığı için bu bölümde yalnızca elde edilen sonuçlar verilmiştir. Ayrıca test dizilerinin uzunluklarına bağlı olarak kapsam kriteri de dikkate alınmıştır. Test dizilerinin önceliklendirme yöntemlerine ve uzunluklarına göre sıralama sonuçları Tablo 3.7'de verilmiştir.

Tablo 3.7. Test dizilerinin önceliklendirme ve sıralama sonuçları

GO		UYÖ		BCO		Uzunluk	
Test Dizileri	***Öncelik Değerleri***	***Test Dizileri***	***Öncelik Değerleri***	***Test Dizileri***	***Öncelik Değerleri***	***Test Dizileri***	***Test Uzunlukları***
TOD1	57233	TOD1	2803	TOD1	2543,852	TOD1	1089
TOD53	10019	TOD53	318	TOD53	310,4476	TOD8	146
TOD37	9996	TOD8	316	TOD37	308,4766	TOD16	146
TOD45	9964	TOD24	315	TOD45	308,4766	TOD24	146
TOD8	9925	TOD37	315	TOD8	308,4169	TOD37	146
TOD24	9885	TOD45	313	TOD24	308,4169	TOD45	146
TOD16	9830	TOD16	312	TOD16	306,4459	TOD53	146
TOD65	6454	TOD65	199	TOD65	188,5699	TOD65	86
TOD81	6449	TOD81	199	TOD81	188,5699	TOD73	86
TOD73	6419	TOD73	198	TOD73	187,5802	TOD81	86
TOD59	5046	TOD59	190	TOD59	168,2257	TOD30	77
TOD30	4935	TOD30	186	TOD30	166,2022	TOD59	76
TOD66	3578	TOD66	115	TOD74	116,0409	TOD66	50
TOD74	3573	TOD74	115	TOD66	115,4409	TOD74	50
TOD82	3572	TOD82	115	TOD82	115,4409	TOD82	50
TOD39	1925	TOD68	68	TOD60	61,7623	TOD32	38
TOD47	1925	TOD76	68	TOD39	60,0242	TOD60	34
TOD55	1925	TOD84	68	TOD47	60,0242	TOD3	28
TOD10	1854	TOD60	67	TOD55	60,0242	TOD68	26
TOD18	1854	TOD10	63	TOD10	59,9645	TOD76	26
TOD26	1854	TOD18	63	TOD18	59,9645	TOD84	26
TOD68	1853	TOD26	63	TOD26	59,9645	TOD10	24
TOD76	1849	TOD39	63	TOD68	59,605	TOD18	24
TOD84	1848	TOD47	63	TOD76	59,605	TOD26	24
TOD67	1626	TOD55	63	TOD84	59,605	TOD39	24
TOD75	1625	TOD32	59	TOD67	53,672	TOD47	24
TOD83	1622	TOD31	56	TOD75	53,672	TOD31	23
TOD60	1601	TOD67	51	TOD83	53,672	TOD67	22
TOD32	1445	TOD75	51	TOD32	52,4808	TOD75	22
TOD31	1282	TOD83	51	TOD31	51,6879	TOD83	22
TOD38	1172	TOD61	39	TOD38	34,6841	TOD61	20
TOD46	1172	TOD9	38	TOD46	34,6841	TOD9	16
TOD54	1172	TOD17	38	TOD54	34,6841	TOD17	16
TOD40	1106	TOD25	38	TOD9	34,6244	TOD25	16
TOD48	1106	TOD38	38	TOD17	34,6244	TOD38	16
TOD56	1106	TOD46	38	TOD25	34,6244	TOD46	16
TOD9	1101	TOD54	38	TOD40	33,1085	TOD54	16
TOD17	1101	TOD19	33	TOD48	33,1085	TOD11	14
TOD25	1101	TOD27	33	TOD56	33,1085	TOD19	14
TOD19	1035	TOD40	33	TOD19	33,0488	TOD27	14
TOD27	1035	TOD48	33	TOD27	33,0488	TOD40	14
TOD3	937	TOD56	33	TOD61	32,1819	TOD48	14
TOD61	774	TOD3	30	TOD3	29,1126	TOD55	14
TOD69	304	TOD69	12	TOD69	10,9143	TOD56	14
TOD77	296	TOD77	12	TOD77	10,9143	TOD69	10
TOD85	291	TOD85	12	TOD85	10,9143	TOD77	10
TOD62	283	TOD62	10	TOD70	8,8949	TOD85	10
TOD78	224	TOD70	10	TOD86	8,8949	TOD62	8
TOD70	221	TOD78	10	TOD78	8,8949	TOD70	8

Tablo 3.7. (Devamı…)

GO		UYÖ		BCO		Uzunluk	
Test Dizileri	***Öncelik Değerleri***	***Test Dizileri***	***Öncelik Değerleri***	***Test Dizileri***	***Öncelik Değerleri***	***Test Dizileri***	***Test Uzunlukları***
TOD86	221	TOD86	10	TOD62	8,811	TOD78	8
TOD63	205	TOD63	8	TOD71	6,9253	TOD86	8
TOD34	160	TOD71	8	TOD79	6,9253	TOD63	6
TOD35	159	TOD79	8	TOD87	6,9253	TOD71	6
TOD33	158	TOD87	8	TOD63	6,8674	TOD79	6
TOD36	157	TOD64	6	TOD72	4,9343	TOD87	6
TOD41	156	TOD72	6	TOD80	4,9343	TOD64	4
TOD49	155	TOD80	6	TOD88	4,9343	TOD72	4
TOD87	155	TOD88	6	TOD64	4,9097	TOD80	4
TOD57	154	TOD4	5	TOD41	4,723	TOD88	4
TOD50	153	TOD5	5	TOD49	4,723	TOD4	3
TOD58	152	TOD6	5	TOD57	4,723	TOD5	3
TOD42	151	TOD7	5	TOD43	4,7199	TOD6	3
TOD44	147	TOD12	5	TOD51	4,7199	TOD7	3
TOD71	147	TOD13	5	TOD44	4,7181	TOD12	3
TOD52	146	TOD14	5	TOD52	4,7181	TOD13	3
TOD43	143	TOD15	5	TOD42	4,715	TOD14	3
TOD51	139	TOD20	5	TOD50	4,715	TOD15	3
TOD79	134	TOD21	5	TOD58	4,715	TOD20	3
TOD64	129	TOD22	5	TOD36	4,7058	TOD21	3
TOD72	109	TOD23	5	TOD35	4,7047	TOD22	3
TOD80	107	TOD28	5	TOD33	4,7026	TOD23	3
TOD5	89	TOD29	5	TOD34	4,7009	TOD28	3
TOD88	89	TOD33	5	TOD12	4,6633	TOD29	3
TOD6	88	TOD34	5	TOD20	4,6633	TOD33	3
TOD4	87	TOD35	5	TOD28	4,6633	TOD34	3
TOD7	86	TOD36	5	TOD14	4,6602	TOD35	3
TOD12	85	TOD41	5	TOD22	4,6602	TOD36	3
TOD20	84	TOD42	5	TOD15	4,6584	TOD41	3
TOD28	83	TOD43	5	TOD23	4,6584	TOD42	3
TOD21	82	TOD44	5	TOD13	4,6553	TOD43	3
TOD29	81	TOD49	5	TOD21	4,6553	TOD44	3
TOD13	80	TOD50	5	TOD29	4,6553	TOD49	3
TOD15	76	TOD51	5	TOD7	4,6461	TOD50	3
TOD23	75	TOD52	5	TOD6	4,645	TOD51	3
TOD14	72	TOD57	5	TOD4	4,6429	TOD52	3
TOD22	68	TOD58	5	TOD5	4,6412	TOD57	3
TOD89	59	TOD2	4	TOD2	3,6656	TOD58	3
TOD2	51	TOD11	4	TOD11	3,6656	TOD2	2
TOD11	51	TOD89	4	TOD89	2,9487	TOD89	2

3.2.3. Test önceliklendirme yöntemlerinin başarımlarının karşılaştırılması

Test dizilerinin hata yakalama başarımları, test önceliklendirme metotlarını karşılaştırmak için önemli bir kriterdir. Test sürecinde ortaya çıkarılan hatalar düzeltilerek test işlemine devam edilir. Bu nedenle test dizilerinin hata yakalama başarımı test sıralamasına göre değişebilir. Yani ilk test edilen dizide yakalanan hata aynı nedenle daha sonraki test dizilerinde ortaya çıkmaz. Hata meydana geldikten

sonra test işlemine baştan başlanacağı için test dizisi içerisinde hatadan sonraki olayların işlenmesi de söz konusu hata düzeltilmeden önce mümkün olmaz. Dolayısıyla test dizilerinin test sırasının değişmesi hata yakalama durumlarını da değiştirecektir. Tüm test dizileri test edilmeden önce sistemde ne kadar hata ortaya çıkacağı da bilinemez. Ancak sisteme belli sayıda hata eklenerek test sürecinde bu hataların ne kadarının yakalandığı ve önceliklendirme metotlarının başarımlarının ne kadar olduğu incelenebilir.

Bu çalışmada sisteme hata eklenmesi yolu izlenmiştir. Burada dikkat edilmesi gereken nokta, hataların önceliklendirme sürecinin bir parçası olmadığı yalnızca test önceliklendirme metotlarının performanslarını belirlemek için kullanıldığıdır. Bu şekilde programcılar tarafından eklenen hataların sayısı ve hangi durumlarda ortaya çıkacağı bilindiği için her bir test dizisinin hata yakalama kapasitesi önceden belirlenebilir. Test takımının her bir test dizisinin hata yakalama kapasitesine göre büyükten küçüğe göre sıralanmasıyla önceliklendirme performanslarını değerlendirmek için bir ideal sıralamalar elde edilir.

3.2.4. Sonuçlar

Model tabanlı test önceliklendirme metotlarının başarımlarını değerlendirmek için genellikle "kapsam" (*ing. coverage*) kriterleri kullanılmaktadır. Kapsam kriterlerine göre ilk test edilecek test dizilerinin nod/olay (*ing. node coverage*) ve ark kapsamının (*ing. arc coverage*) geniş olması beklenir. İncelenen testlerin ark ve nod kapsamlarının geniş olması sistemin mümkün olduğunca çok kısmının kontrol edildiği anlamına gelirken uzun test dizilerinin işletilmesi test maliyetini de artırır. Maliyet kısıtı dikkate alındığında ise kısa test dizilerini seçme eğilimi görülmektedir. Ancak hata yakalama başarımları da, test maliyetini düşürmek kadar önemlidir. Çoğu zaman test dizilerini hata yakalama başarıları diziyi oluşturan olaylara bağlıdır. Mevcut öncekliklendirme kriterleri test dizilerini bir bütün olarak ele alıp dizilere ilişkin bilgilerle önceliklendirme yaptığından diziyi oluşturan olayları ihmal etmektedir. Bu çalışmada ise olaylara ve olayların sistem için önemine odaklanılmıştır. Böylece test dizilerinin kapsamı yanında dizinin kapsadığı olayların sistem için önemi ve bir olayın hata yakalaması durumunda bu hatayı düzeltmek için gerekli maliyetler dikkate alınarak en uygun çözüme ulaşmak amaçlanmıştır.

Kümeleme ile test önceliklendirme yöntemiyle farklı olaylardan oluşan ancak eşit sayıda olay kapsayan test dizileri arasında kapsadıkları olayların önemlilik dereceleri doğrultusunda farklı sıralamalar yapmak mümkün olmuştur. Hatta önemlilik derecesi yüksek olayları kapsayan nispeten kısa test dizilerinin öncelik sıralarının yüksek olmasıyla daha az sayısa test ve daha az tıklama ile sistemdeki hataların ortaya çıkarılmasıyla test maliyetini düşürmek mümkün olmuştur.

Kümeleme ile test önceliklendirme metotlarının başarımlarının değerlendirilmesi için 4 tane değerlendirme kriteri önerilmiştir. Bu kriterler sistem testi sonucunda elde edilen hata sayıları kullanılarak tanımlanmıştır. Her test dizisinin birbirinden bağımsız olarak ortaya çıkardığı hata sayısı ve ya hataların ortaya çıktığı olayların ne kadarını içerdiklerini dikkate alan bu kriterler aşağıda açıklanmıştır.

1. ***Hata yakalayan olay kapsama kriteri (HYOK):*** Test sürecinde, hata ile karşılaşılan olaylar tespit edilmiş ve bu olayları içeren test dizileri kapsadıkları hata yakalayan olay sayısına göre büyükten küçüğe doğru sıralanmıştır. Bu sıralama HYOK için en iyi önceliklendirme performansına sahip ideal sıralama olarak kabul edilmiştir. Daha sonra kümeleme ile önceliklendirme metotlarıyla elde edilen sıralamalar üzerinden testlerinin HYOK kapasiteleri ideal sıralama ile karşılaştırılmıştır.
2. ***Tekrarlı hata yakalama kriteri (TRHY):*** Bir test dizisinde aynı olayın birden çok işletilmesi gerekebilir ve bu olayların her biri aynı hatayı yakalama potansiyeline sahiptir ancak hata ilk kez karşılaşılan olay tarafından ortaya çıkarılır. Bunun yanısıra bazı özel durumlarda ikinci veya daha sonraki karşılaşmalarda da hata meydana gelebilir. Yani bir hatanın ilk kez ortaya çıkması yada tekrarlaması önceki işlem adımlarına bağlı olabilir. Bu nedenle test dizileri, tekrarlı hata sayıları dikkate alınarak sıralandığında ideal hata yakalama sayıları elde edilir. İdeal sıralama kullanılarak kümeleme ile test önceliklendirme metotları ile elde edilen test sıralamalarının tekrarlı hata yakalama potansiyelleri karşılaştırılmıştır.
3. ***Tekrarsız hata yakalama kriteri (TZHY):*** Bu kriter gerçek test sürecinde olduğu gibi karşılaşılan hataların düzeltildiği ve aynı hataya bir test dizisinde aynı nedenle bir daha karşılaşılmadığı durumu ifade eder. Başarım değerlendirmesinde tekrarsız hata sayılarına göre test dizileri büyükten küçüğe

doğru sıralanarak elde edilen ideal sıralama test önceliklendirme metotlarının başarımlarını karşılaştırmak için temel alınmıştır.

4. ***Tüm hataları ortaya çıkarma hızı (HOÇH):*** Son olarak da önerilen metotlardaki sıralar kullanılarak sistemde ortaya çıkarılan tüm hataları yakalamak için gerekli test sayısı açısından metotlar karşılaştırılmıştır.

Test sonucunda özel indirimler modülünde 50 hata ile karşılaşılmıştır. Hata listesi ve hataların ortaya çıkarıldığı olaylar Tablo 3.8'de verilmiştir.

Tablo 3.8. Test sonucunda ortaya çıkarılan hatalar

Hata No	***Hata Yakalanan Olaylar***	***Hata Açıklaması***
1	ADDE1	ADD tuşuna bir kez basınca bir daha aktif olmuyor.
2	ADDE1	ADD tuşuna basınca girilen tüm özel indirim tanımlama bilgileri siliniyor ve ADD tuşu pasif hale geliyor.
3	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim tanımlama eklemiyor ve girilen bilgiler ekranda kalıyor.
4	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim eklemiyor
5	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim eklemiyor
6	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim eklemiyor ve "Data could not be saved " hatası veriyor
7	ADDE1	Özel indirim tanımlama bilgilerinin hiç birini girmeye izin vermiyor "A maximum of 10 specials I allowed" hatası veriyor
8	ADDE1	Özel indirim tanımlama bilgilerinin hiç birini girmeye izin vermiyor "A maximum of 10 specials I allowed" hatası veriyor
9	ADDE1	Özel indirimlerin son bulacağı tarih bilgisini girmeden de tanımlama yapılmasına izin veriyor
10	ADDE1	Doldurulması zorunlu tüm alanlara veri girişi yapıldığında özel fırsat tanımlaması yapıyor fakat DELETE ve EDIT tuşları yerine "in process" yazısı geliyor
11	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim eklemiyor "This functionality is not fully programmed" ve "Data could not be saved" hatalarını veriyor
12	ADDE1	Doldurulması zorunlu tüm alanlara veri girilmiş olsa bile yeni bir özel indirim eklemiyor "This functionality is not fully programmed" ve "Data could not be saved" hatalarını veriyor
13	ADDE1	"Add specials to list" ekranında resim eklemiyor ama "edit"te ekliyor. Resim eklenmese bile tanımlanan özel indirim türünün yanında resim simgesi görünüyor
14	ADDE2	Birden fazla özel inridim tanımlanmasına izin verilmiyor. "Maximum specials of 10 allowed" hatası veriyor.
15	ADDE2	Aynı isimle özel indirim tanımlanmasına izin veriliyor ancak yine de "name already exist" hatası veriyor

Tablo 3.8. (Devamı…)

Hata No	Hata Yakalanan Olaylar	Hata Açıklaması
16	ADDE2	Aynı isimle özel indiri tanımlanmasına izin veriliyor
17	OKDEL2	DELETE tuşuna basıldığında tüm özel indirimler siliniyor ve ADD tuşu aktif olmuyor
18	OKDEL2	Özel indirimi silmiyor ve "error while deleting" hatasını veriyor
19	OKDEL2	Özel indirimi silmiyor ve "error while deleting" hatasını veriyor
20	OKDEL2	Özel indirimi silmiyor ve "error while deleting" hatasını veriyor
21	OKDEL2	Silmek için hangi özel indirimi seçilirse seçilsin ilk özel indirim siliniyor
22	EDIT2	EDIT tuşuna basıldığında tüm özel indirimleri siliyor ve ADD tuşu aktif olmuyor
23	EDIT2	EDIT tuşuna basıldığında edit ekranı gelmiyor, ekran aynı kalıyor
24	EDIT2	EDIT tuşuna basıldığında edit ekranı gelmiyor, ekran aynı kalıyor
25	EDIT2	Düzenlemek için hangi özel indirimin edit tuşuna tıklanırsa tıklansın ilk özel indirimin edit ekranı geliyor.
26	EDIT2	"Fatal Error: Call to a member function as Array()" hatasını veriyor
27	SAVE2	EDIT'ten sonta SAVE tuşuna tıklandığında tüm özel indirimleri siliyor ve ADD tuşu aktif olmuyor
28	SAVE2	EDIT'ten sonra SAVE tuşuna tıklandığında özel indirimler menüsünde bazı bilgiler dolu kalıyor
29	SAVE2	SAVE tuşuna tıklandığında "Add to special to list" ekranı gelmiyor edit ekranında kalıyor
30	SAVE2	EDIT'ten sonra SAVE tuşuna tıklandığında "Data could not be saved" hatasını veriyor
31	SAVE2	EDIT'ten sonra SAVE tuşuna tıklandığında "special name exists" fonksiyonunda hata veriyor. Fatal Error
32	SAVE2	Özel indirim tanımlamalarında resim eklendiğinde yada edit menüsünde resim eklendiğinde hata veriyor ve resmi eklemiyor. "This functionality is not fully programmed" ve "Data could not be saved" hatalarını veriyor
33	SAVE2	Özel indirim eklemiyor. "Data could not be saved" hatalarını veriyor
34	CANE2	EDIT'ten sonta CANCEL tuşuna tıklandığında tüm özel indirimleri siliyor ve ADD tuşu aktif olmuyor
35	SAVE2	Edit bölümünde değişiklik yapıldığı halde değişiklik kabul edilmiyor.
36	SAVE2	Resim silindiğinde tüm özel indirim tanımlamaları siliniyor
37	CANE2	Resim sil (PHOTO DELETE) tuşuna tıklandığında çıkan pencerede CANCEL tuşuna tıklansa bile resmi siliyor.
38	SAVEINC2	Edit ile değişiklik yapıldıktan sonra eksik veri olsa bile kayıt yapıyor.
39	SAVEINC2	Edit menüsünden değişiklik yapıldıktan sonra eksik veri olduğu durumda bile kaydedebiliyor.
40	ADDE3	Resim eklemiyor
41	ADDINC1	Özel indirimlerin geçerli olduğu tarihler girilmeden tanımlama yapılmasına izin veriliyor ve tarihler girilmediğinde aynı isimli özel fırsatlar tanımlanabiliyor.
42	ADDINC1	Başlangıç tarihi girilmese de özel indirim tanımlanabiliyor
43	ADDINC1	Eksik veri ile özel indirim ekleyebiliyor
44	TODAY22	TODAY tuşuna tıklandığında bugünün tarihi aktif olmuyor
45	TODAY21	TODAY tuşuna tıklandığında bugünün tarihi aktif olmuyor
46	OK21	Geçmiş tarihli kayıt yapılmasına izin veriyor
47	OK22	Geçmiş tarihli kayıt yapılmasına izin veriyor
48	OK21	Geçmiş tarihli özel indirim tanımlanıyor ve hata mesajı vermiyor.
49	OK22	Geçmiş tarihli özel indirim tanımlanıyor ve hata mesajı vermiyor.
50	OK22	Özel indirim tanımlamalarında çakışan tarihler eklenebiliyor ve hata mesajı vermiyor.

3.2.4.1. Hata yakalayan olay kapsama (HYOK) kriteri

Tablo 4.8'de verilen hatalar ve bu hataların ortaya çıktığı olaylar, test sırasında ilk kez karşılaşılan hatalardır ve bu hataların tamamı ilk karşılaşıldıkları durumda düzeltilerek bir sonraki adıma geçilmiştir. Ancak test dizileri içinde aynı fonksiyonu kullanan farklı isimlerle adlandırılmış olaylar bulunmaktadır. Örneğin TODAY11, TODAY12, TODAY22, ve TODAY21 olayları today butonuna basılması durumunda çalıştırılacak olaylardır. Aralarındaki farklılık yalnızca olayların çağrılma yeridir. Dolayısıyla TODAY22 olayının yakaladığı bir hata test dizilerinin işletilme sırasına göre ilk olarak TODAY11'in işletildiği test dizilerinde de ortaya çıkarılabilecek bir hatadır. Bu nedenle hata yakalayan olay kapsam kriterine göre TODAY11'de hata ile kaşılaşılan bir olay olarak kabul edilir. Aynı şekilde TODAY12, TODAY22 ve TODAY21 ayrı ayrı hata yakalama noktaları olarak ele alınmıştır. Bu kritere göre test dizileri en fazla hata ile karşılaşılabilecek test dizisinden başlayarak büyükten küçüğe göre sıralanmış ve bu sıralama ideal sıralama olarak kabul edilmiştir. İdeal sıralama da önerilen test önceliklendirme metotlarının başarımlarını karşılaştırmak için kullanılmıştır. Ek 3.4'de Tablo E3.2'de her bir test dizisinin kapsadığı hata ortaya çıkaran olay sayısı verilmiştir. Bu kritere göre değerlendirme yapılırken test dizileri birbirinden bağımsız olarak ele alınmış ve bir test dizilerinin önceki dizilerde ortaya çıkarılan hataların düzeltilmesinden etkilenmedikleri varsayılmıştır.

Bu sonuçların istatistiksel değerlendirilmesi için Kendall Tau testi kullanılmıştır. HYOK kriterine göre oluşturulan ideal test sırası ile önceliklendirme ve uzunluk sıralamasıyla elde edilen sıralamalar tek tek karşılaştırılmış ve aralarındaki ilişki 0,01 anlamlılık düzeyinde test edilmiştir. İlişki analizleri için kullanılan Kendall Tau testinin hipotezleri şu şekilde kurulur:

H_0: İdeal sıralama ile önceliklendirilmiş sıra arasında anlamlı bir ilişki yoktur.
H_1: İdeal sıralama ile önceliklendirilmiş sıra arasında anlamlı bir ilişki vardır.

Sıfır hipotezi (H_0)'nin reddedilmesi önerilen önceliklendirme metodu ile ideal sıralama arasındaki ilişkinin 0,01 seviyesinde anlamlı olduğunu ifade eder. Test

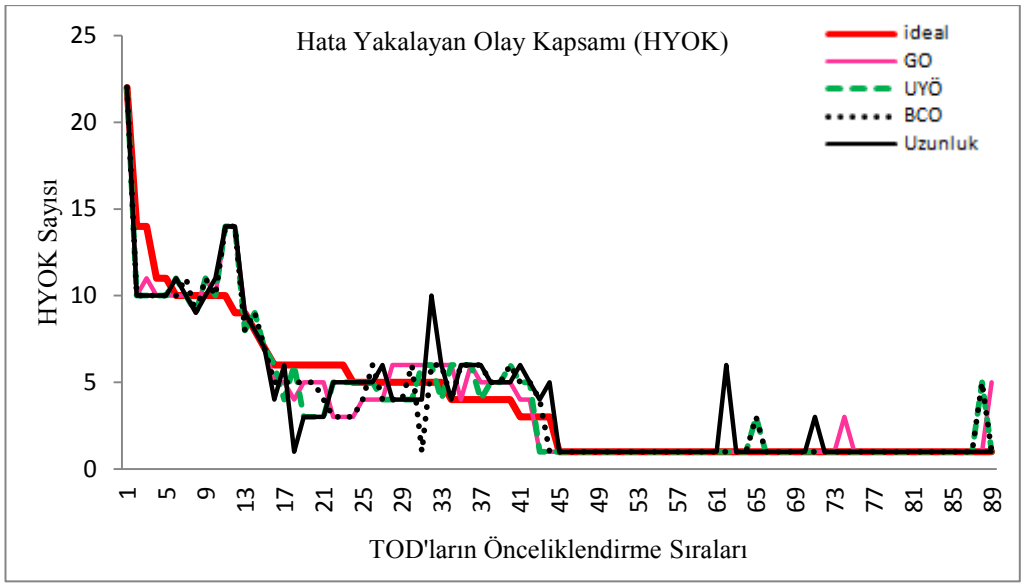
sonucunda hesaplanan Kendall Tau korelasyon katsayısının değeri ise ilişkinin yönünü ve kuvvetini verir.

Tablo 3.9'da verilen sonuçlar, HYOK kriteri için gerek önceliklendime metotları gerek uzunluğa göre sıralama ile ideal durum arasında pozitif yönlü ve yüksek bir korelasyon olduğu görülmektedir. Hesaplanan tüm korelasyon katsayıları anlamlıdır. Önceliklendirme metotlarının başarımı artırdığı görülmektedir ve bunlar arasında en güçlü korelasyona sahip olan UYÖ'dır.

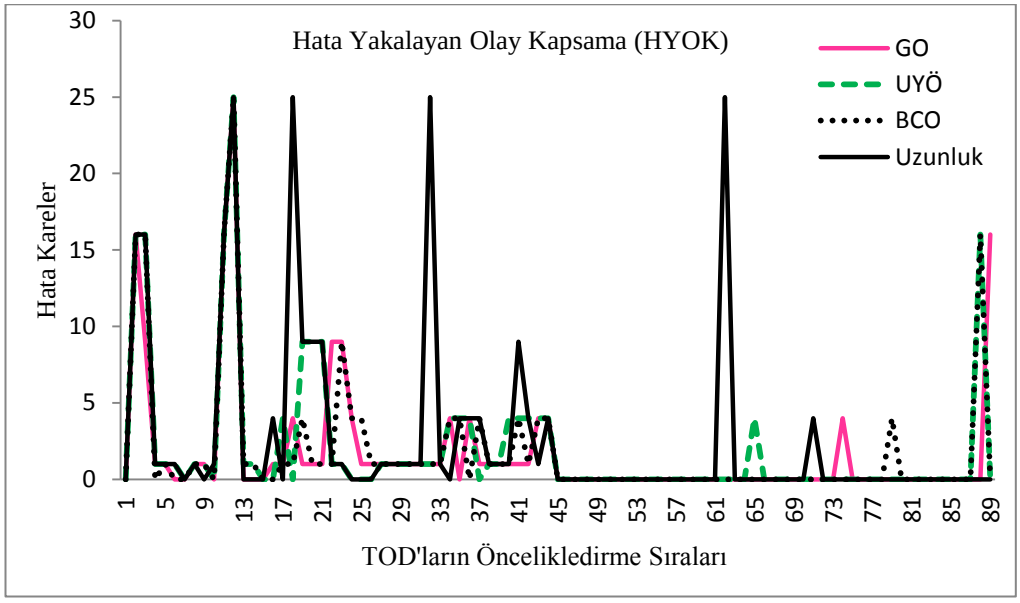
Tablo 3.9. HYOK kriterine göre ideal durum ile önceliklendirme metotları arasındaki ilişki analizi sonuçları

			UYÖ	GO	BCO	Uzunluk
Kendall's Tau_b	İdeal	Korelasyon katsayısı	0,839	0,816	0,837	0,795
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000
	UYÖ	Korelasyon katsayısı		0,846	0,889	0,819
		Anlamlılık (çift-taraflı)		0,000	0,000	0,000
	GO	Korelasyon katsayısı			0,900	0,796
		Anlamlılık (çift-taraflı)			0,000	0,000
	BCO	Korelasyon katsayısı				0,808
		Anlamlılık (çift-taraflı)				0,000

Önceliklendirilmiş ve uzunluğu göre sıralanmış test dizilerinin HYOK değerlerinin ideal sıralama ile karşılaştırmaları Şekil 3.7'deki grafikte verilmiştir. Grafikte yatay eksen test öncelik sırasını gösterirken dikey eksen her bir test dizisi için hata ile karşılaşılan olay sayısını temsil etmektedir. İdeal sıranın kırmızı çizgi ile temsil edildiği bu grafikte her üç metodunda bazı küçük sapmalarla ideal sıraya yakın bir grafik çizdiği görülmektedir. Ancak yalnızca uzunluk dikkate alındığında eşit uzunluktaki test dizileri arasında ayrım yapmak mümkün olmadığından rasgele sıralama yapılmaktadır. Grafikte de görüldüğü gibi ilk sıralarda incelenen testlerin daha çok hata yakalayan olay içermektedir. Bu da önerilen önceliklendirme metotlarının başarısının beklenen başarıya yakın olduğunu göstermektedir.



Şekil 3.7. Test önceliklendirme metotlarının HYOK kriterine göre ideal sıralama ile karşılaştırmalı grafiği

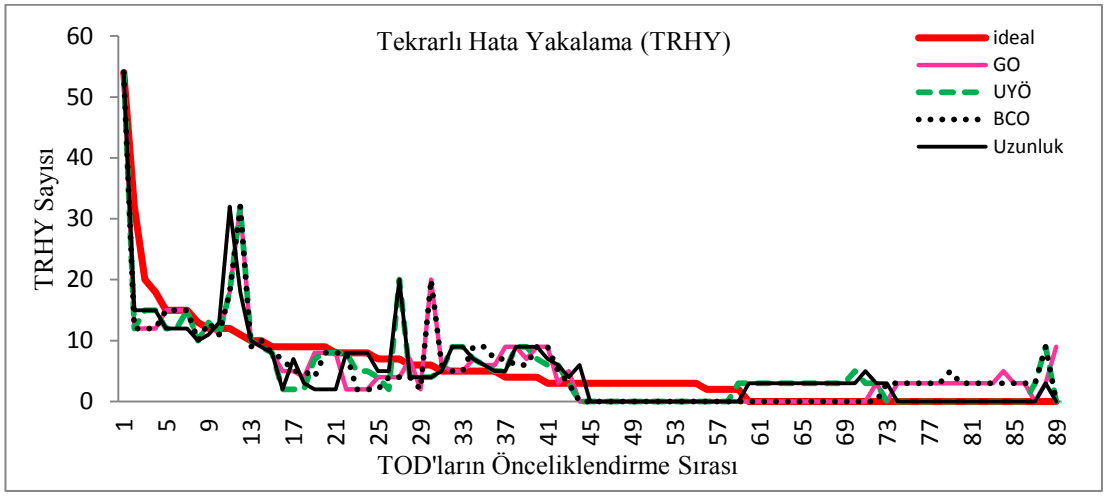


Şekil 3.8. Test önceliklendirme metotlarının HYOK kriterine göre ideal sıralamadan sapma grafiği

Şekil 3.8'de dört durumda elde edilen sıralamaların ideal durumdan sapmalarının grafiği verilmiştir. Grafikte yatay eksen test dizilerinin öncelik sırasını dikey eksen ise hata idealden sapmaların karesinin yani hata kareleri göstermektedir. Bu grafikde uzunluğa göre yapılan sıralamanın HYOK kriteri için idealden daha fazla sapmaya sahip olduğu görülmektedir.

3.2.4.2. Tekrarlı hata yakalama (TRHY) kriteri

Test sırasında hata yakalanan olaylar dikkate alınarak, bu olayların hata üretme potansiyelleri belirlenmiştir. Bu kriter içinde hataların ilk ortaya çıktığında düzeltildiği ihmal edilmiş, hata üreten olayın her kullanıldığında aynı hatayı ürettiği varsayılmıştır. Böylece test dizilerinin birbirinden bağımsız olarak kaç tane hata üretebilme kapasitesine sahip olduğu dikkate alınmıştır. Dolayısıyla her testin tekrarlı hata yakalayabilme potansiyeli belirlenerek testler en yüksek kapasiteye sahip testten en düşüğe doğru sıralanmıştır. İdeal olarak kabul edilen bu sıralama ile önceliklendirme metotlarıyla elde edilen sıralamalara göre testlerin tekrarlı hata yakalama kapasiteleri Ek 3.4'de Tablo E3.3'de verilmiştir. Test önceliklendirme metotlarının TRHY kriterine göre ideal sıralama ile karşılaştırmalı grafiği Şekil 3.9'da verilmiştir.

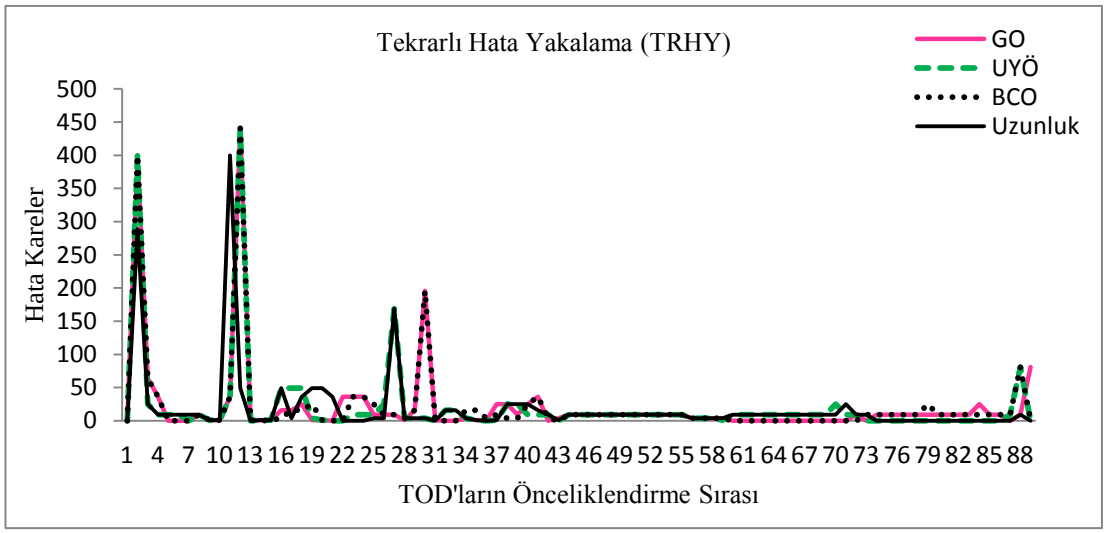


Şekil 3.9. Test önceliklendirme metotlarının TRHY kriterine göre ideal sıralama ile karşılaştırmalı grafiği

Grafikte söz konusu test önceliklendirme metotlarının ideal sıralamayla benzer bir eğilim gösterdiği görülmektedir. İstatistiksel olarak ideal sıralama ile önceliklendirme metotları arasındaki ilişkinin yönü ve kuvveti Kendall Tau testi kullanılarak test edilmiştir. Tablo 3.10'da verilen sonuçlara göre her üç metotda ideal sıralama ile kuvvetli bir korelasyna sahiptir ancak 0,823 korelasyon ile en güçlü ilişki BCO ile elde edilen sıralama arasındadır. Ayrıca Şekil 3.10'da TRHY kriterine göre önceliklendirme metotlarının ideal durumdan sapmalarının grafiği verilmiştir.

Tablo 3.10. TRHY kriterine göre ideal durum ile metotlar arasındaki ilişki analizi sonuçları

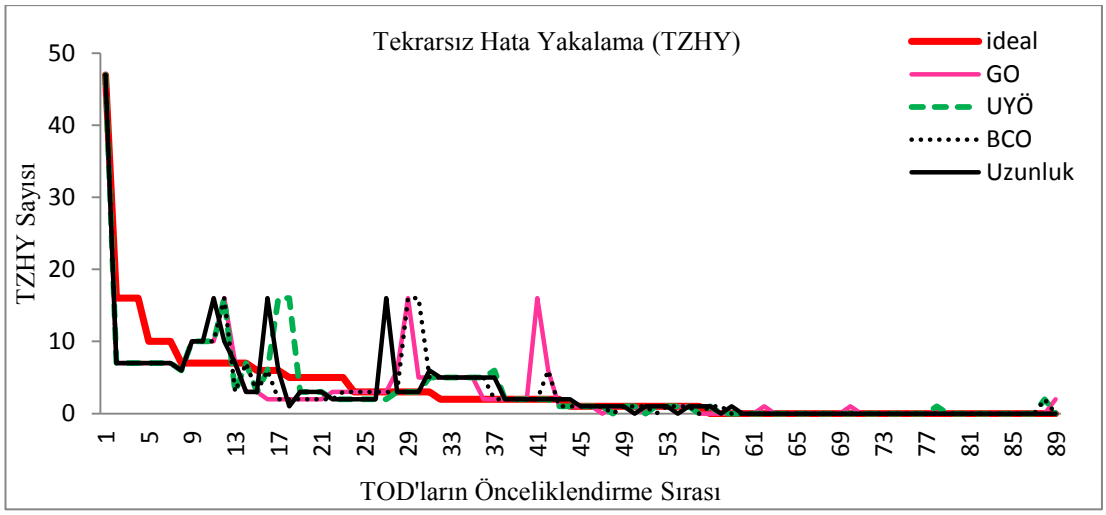
			UYÖ	GO	BCO	Uzunluk
Kendall's Tau_b	İdeal	Korelasyon katsayısı	0,807	0,792	0,823	0,821
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000
	UYÖ	Korelasyon katsayısı		0,802	0,899	0,828
		Anlamlılık (çift-taraflı)		0,000	0,000	0,000
	GO	Korelasyon katsayısı			0,827	0,790
		Anlamlılık (çift-taraflı)			0,000	0,000
	BCO	Korelasyon katsayısı				0,830
		Anlamlılık (çift-taraflı)				0,000



Şekil 3.10. Test önceliklendirme metotlarının TRHY kriterine göre ideal sıralamadan sapmalarının grafiği

3.2.4.3. Tekrarsız hata yakalama (TZHY) kriteri

TZHY kriteri, her testin birbirinden bağımsız olarak test edilmesi durumunda kaç tane hata yakalayabileceğini ifade etmektedir. Bir test dizisi işletilmeye başladığında karşılaşılan hataların düzeltilmesi ve farklı bir neden olmaksızın bir hata ile bir daha karşılaşılmadığı durumları dikkate almaktadır. Bu kritere göre ideal test sıralaması oluşturulmuş ve test önceliklendirme metotlarıyla elde edilen sıralamalarla karşılaştırılarak metotların performans değerlendirmesi yapılmıştır. TZHY başarımları Ek 3.4'de Tablo E3.4'de verilmiştir.

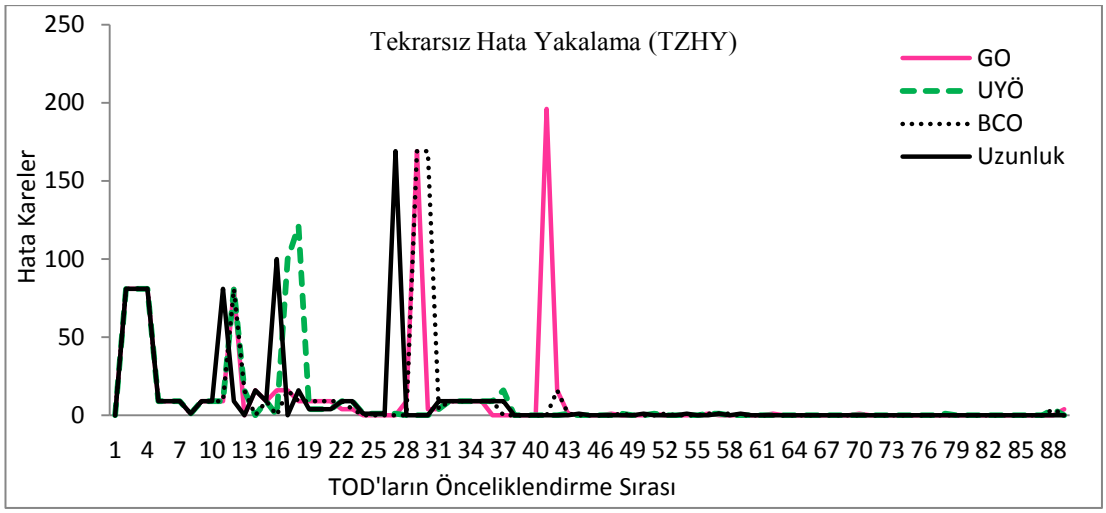


Şekil 3.11. Test önceliklendirme metotlarının TZHY kapasitesine göre ideal sıralama ile karşılaştırmalı grafiği

Tablo 3.11. TZHY kriterine göre ideal durum ile önceliklendirme metotları arasındaki ilişki analizi sonuçları

			UYÖ	GO	BCO	Uzunluk
Kendall's Tau_b	İdeal	Korelasyon katsayısı	0,824	0,790	0,800	0,817
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000
	UYÖ	Korelasyon katsayısı		0,780	0,863	0,864
		Anlamlılık (çift-taraflı)		0,000	0,000	0,000
	GO	Korelasyon katsayısı			0,844	0,775
		Anlamlılık (çift-taraflı)			0,000	0,000
	BCO	Korelasyon katsayısı				0,861
		Anlamlılık (çift-taraflı)				0,000

Şekil 3.11'de görülen grafikte test önceliklendirme metotlarının testleri TZHY kapasitesine göre gerçekleştirilen ideal sıralamayla karşılaştırılmıştır. Önceliklendirme metotları ile ideal sıralama arasında doğrusal ve güçlü bir ilişkinin varlığı grafikte görülmektedir ancak ilişkinin derecesini ve kuvvetini belirlemek için istatistiksel olarak Kendall Tau testi uygulanmış ve sonuçlar Tablo 3.11'de verilmiştir. Bu kritere göre ideal duruma en yakın sıralama ile UYÖ ile elde edilen sıralamadır. 0,824 korelasyon katsayısı doğrusal ve güçlü bir ilişkiyi temsil etmektedir. 0,000 anlamlılık değeri bu korelasyon katsayısının istatistiksel olarak anlamlı olduğunu ve elde edilen bu sonuçların tesadüfi olmadığını ifade eder. Ayrıca Şekil 3.12'de idealden sapma grafiğinde UYÖ ile önceliklendirmenin başarımı daha net görülmektedir. Özellikle 27-31 ve 40-43 aralığındaki testlerde BCO ve GO'nın idealden sapması yüksekken UYÖ sıfır noktasındadır.



Şekil 3.12. Test önceliklendirme metotlarının TZHY kapasitesine göre idealden sapma grafiği

3.2.4.4. Tüm hataları ortaya çıkarma hızı (HOÇH)

Test sürecinde test takımının tamamının test edilmesiyle sistemde toplam 50 hata ortaya çıkarılmıştır. HOÇH kriteri ile hataların ilk karşılaşıldığı anda düzeltildiği ve birbirini takip eden test dizilerinde aynı hata ile aynı nedenden dolayı bir daha karşılaşılmadığı gerçek bir test süreci dikkate alınmıştır. Bu uygulamada ele alınan test dizileri içinde TOD_1, 1089 olay uzunluğuyla kapsam genişliği en yüksek olan test dizisidir ve her üç metot içinde 1. öncelik derecesine sahiptir. Dolayısıyla üç metot içinde ilk test edilecek test dizisi olarak belirlenmiştir. Bu nedenle de gerçek test sürecinde de ilk olarak TOD_1 test edilmiş ve 47 tane hata yakalanmıştır. Sonraki test dizilerinin çalıştırılmasıyla da yalnızca 3 yeni hata yakalanabilmiştir. Bu 3 hatada TOD_8'in testi sırasında ortaya çıkmıştır. Bu kriter dikkate alınarak Tablo 3.7'de verilen test sıralamaları incelendiğinde tüm hataları ortaya çıkarma hızı en yüksek metodun UYÖ olduğu görülmüştür. Test sürecinde UYÖ ile elde edilen sıralama kullanıldığında 3 test dizisi test edilmesiyle tüm hatalar ortaya çıkarılmıştır. Diğer test dizilerinin çalıştırılması sırasında yeni hata ile karşılaşılmamamıştır. BCO ve GO ile sıralamada ilk 5 test dizisinin çalıştırılmasıyla tüm hatalar ortaya çıkarılmıştır. Test dizilerinin uzunluklarına göre sıralanması durumunda TOD_8 ile eşit uzunlukta 6 test dizisi daha bulunmaktadır. Bunlar arasında önem açısından ayrım yapacak bir bilgi bulunmaması test sırasında bu diziler arasında rasgele bir seçim yapılmasını gerektirir. Bu önemli bir dezavantajdır. Özellikle de eşit uzunlukda test dizilerinden oluşan bir test takımında test maliyeti çok artabilir.

Tablo 3.12. Sistemde karşılaşılan tüm hataların ortaya çıkarılma maliyeti

	UYÖ	GO	BCO
Maliyet	3α+1381β	5 α +1673 β	5 α +1673 β

Bu sonuçlar dikkate alınarak tüm hataları yakalamak için gerekli test maliyetleri Tablo 3.12'de verilmiştir. α ve β katsayıları test bütçesine göre belirlenirler ve α her testin başlatma maliyeti, β ise her bir tıklama maliyetine karşılık gelmektedir. Örneğin, UYÖ ile elde edilen test sıralaması sonucunda bütün hataları ortaya çıkarmanın maliyeti, 3 test dizisi çalıştırıldığından başlangıç maliyetinin 3 katı ve bu test dizilerinin toplam uzunluğu olan 1381 tıklama olarak belirlenmiştir. GO ve BCO ortalamalar 5 test ve toplamda 1673 tıklama maliyeti ile sistemdeki hataları ortaya çıkarmıştır.

Tablo 3.12'de THOÇ criterine gore maliyet değerlendirmesinde test dizilerinin uzunluklarına gore sıralanmasına yer verilmemiştir. Çünkü bu sıralamada 2.'den 7.'e kadar olan test dizilerinin uzunlukları eşittir ve test dizileri arasında test önceliği açısından ayrım yapabilecek bir ön bilgi mevcut değildir. Bu nedenle test için test dizileri arasında rasgele seçim yapılır. Dolayısıyla maliyet değerlendirmesinde rasgele oluşabilecek tüm sıralamalar dikkate alınmalıdır. Eşit uzunluktaki 6 test dizisinin 720 farklı dizilişi sözkonusudur. Bu durumda rasgele seçim yapılacak olası tüm durumlardan yalnızca 1/6 tanesi yani 0,17'si maliyet açısından UYÖ'den daha iyi bir performans gösterirken, 0,17'si ise UYÖ ile aynı performansa sahiptir. Diğer tüm durumlarda tüm hataları ortaya çıkarma maliyeti UYÖ'den daha yüksektir Kısaca test dizileri uzunluklarına gore sıralandığında maliyet açısından UYÖ ile önceliklendirmeden daha kötü bir sonuç elde edilmesi olasılığı 0.67'dir. Özellikle kritik yazılımların testinde bu oldukça önemli bir farklılıktır.

4. TARTIŞMA

Bu çalışmada elde edilen sonuçlar kümeleme ile test önceliklendirme metotlarının hata yakalama başarımını artırdığını göstermiştir. Tablo 4.1'de önceliklendirme metotlarının genel başarımları karşılaştırılmış ve UYÖ ile önceliklendirmenin HYOK, TZHY ve HOÇH kriterlerine göre en iyi başarıma sahip olduğu görülürken TRHY kriteri için ise BCO ile önceliklendirmenin daha başarılı olduğu görülmüştür.

Tablo 4.1. Önceliklendirme metotlarının genel başarım tablosu

		Kendall's Tau_b	UYÖ	GO	BCO	Uzunluk
HYOK	İdeal	Korelasyon katsayısı	**0,839**	0,816	0,837	0,795
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000
TRHY	İdeal	Korelasyon katsayısı	0,807	0,772	**0,823**	0,821
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000
TZHY	İdeal	Korelasyon katsayısı	**0,824**	0,790	0,800	0,817
		Anlamlılık (çift-taraflı)	0,000	0,000	0,000	0,000

Sonuç olarak bu çalışmada kümeleme ile önceliklendirme yönteminin yazılım koduna ve ön bilgi olarak hata bilgisine ihtiyaç duymadan hataları ortaya çıkarmada başarılı olduğu görülmüştür. Ayrıca önerilen yöntemin test sürecinde test dizilerinin uzunluğu yerine sistem bileşenlerinin hata yakalama performanslarını ön plana çıkarması önemli bir avantajdır.

Bu çalışma farklı düzeylerde modelleme ile anlamlı küçük parçalara ayrılmış sistem modellerini birlikte ele alarak, bir yazılımın bütün olarak test edilmesine imkan sağlar. Önerilen önceliklendirme yaklaşımı modellenmiş her hangi bir yazılımın testi sürecinde gerek programcılar gerekse programcılar dışında kişiler tarafından kullanılabilir. Ayrıca bu çalışmada kümelemeye dayalı önceliklendirme yaklaşımı ayrıntılı bir şekilde açıklanmıştır. Örnek olarak da klasik kümeleme ve bulanık kümeleme yöntemlerinden basit ve en sık kullanılan iki algoritma tercih edilmiştir. Ancak önerilen yöntem farklı kümeleme algoritmalarının da önceliklendirme probleminin çözümüne uygulanmasına izin vermektedir.

Yazılım test sürecinde modele dayalı test yaklaşımları kullanıldığında kod bilgisi olmadığından ortaya çıkarılan hataların düzeltilmesi mümkün olmaz. Dolayısıyla bazı hataların düzeltilmemesi durumunda da test dizisinin tamamı çalıştırılamaz. Bu nedenle kümeleme ile önceliklendirme yönteminin kullanıldığı test süreci programcı ve testçilerin işbirliği içinde çalışmasını gerektiren bir süreçtir. Böylece ortaya çıkarılan hatalar kod üzerinde düzeltilerek test süreci sağlıklı ve hızlı bir biçimde tamamlanabilir.

Hata bilgisi olmadan test önceliklendirme başarımlarını ölçmek için kullanılan kapsam yada maliyet kriterleri özellikle eşit uzunlukdaki test dizileri arasında ayrım yapmak için yeterli gelmeyebilir. Kümelemeye dayalı önceliklendirme yöntemleri test dizisini oluşturan olayların önemini ortaya çıkarmayı amaçlar ve önemli olaylardan oluşan test dizilerinin ilk olarak test edilmesini önerir. Dolayısıyla test takımı eşit uzunluktaki tes dizilerinden meydana gelmiş olsa dahi test dizileri önceliklendirilebilir. Bu kapsam kriterini kullanan önceliklendirme yöntemlerine göre önemli bir avantajdır.

Yazılımın kod bilgisini veya önceki çalıştırmalarda ortaya çıkan hata bilgisine başvurmaması bu yöntemin bir dezavantajı olarak görülebilir ancak söz konusu bilgiler mevcut olduğunda ilave faktörlerle veri setine dahil edilebilir. Yani faktör esnekliği kullanılarak kümeleme ile test önceliklendirme yöntemi her iki durum için de genişletilebilir.

Model tabanlı test yöntemlerinin model ve test dizilerindeki değişikliklerden etkilenmesi genel bir sorundur. Model değişince test dizileri ve öncelik değerleri de değişebilir. Ayrıca benzer nedenlerden dolayı bu çalışmada oluşturulan modeller ve modeller üzerinden üretilen test takımının hatasız olduğu varsayılmıştır. Önerilen yöntem hataları ortaya çıkarmada kullanıcı davranışlarını dikkate alır. Bu nedenle program içindeki işlem operatörlerinin yanlış kullanılması sonucu oluşan yanlış hesaplamalar ya da programdaki mantıksal hataları ortaya çıkaramayabilir. Ayrıca bu yöntemin başarısını değerlendirmek için test sonucunda ortaya çıkarılan hata sayısına başvurulması da bir dezavantajdır.

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada kümeleme ile önceliklendirme yönteminin yazılım koduna ve ön bilgi olarak hata bilgisine ihtiyaç duymadan hataları ortaya çıkarmada başarılı olduğu görülmüştür. Kümeleme ile test önceliklendirme yöntemi bir yazılımın bileşenleri olan olaylara ve olayların sistem için önemine odaklanmıştır. Kümeleme ile test önceliklendirme yöntemiyle farklı olaylardan oluşan ancak eşit sayıda olay kapsayan test dizileri arasında kapsadıkları olayların önemlilik dereceleri doğrultusunda farklı sıralamalar yapmak mümkün olmuştur. Hatta önemlilik derecesi yüksek olayları kapsayan nispeten kısa test dizilerinin öncelik sıralarının yüksek olmasıyla daha az sayısa test ve daha az tıklama ile sistemdeki hataların ortaya çıkarılmasıyla test maliyeti düşürülebilir. Ayrıca sistmee ait ön bilgiye ihtiyaç duymamaıs nedeniyle yazılım geliştirme sürecinde regresyon testlerinden farklı olarak kodlamadan sonraki her aşamada kullanılabilir.

Gelecek çalışmalarda tartışma bölümünde bahsedilen dezavantajları ortadan kaldırmak amacıyla yöntemin şu şekilde geliştirilmesi planlanmaktadır:

- UYÖ ve BCO algrotimaları küme merkezi olarak nokta ptototipini kullanırlar ve küme merkezine yakın olan olayları önemli kabul ederler. Gelecek çalışmalarda farklı küme prototipleri kullanılarak önceliklendirme performansının artırılması planlanmaktadır.
- Bulanık kümelemenin bu çalışmada ele alınan uygulamalar da olduğu gibi gerçek problemlerin çözümünde klasik kümelemeye göre daha etkili sonuçlar üretmesi beklenirken bu çalışmada elde edilen sonuçlar klasik mantığa dayalı UYÖ ile önceliklendirmenin hata yakalama ve hataları erken ortaya çıkarma konusunda daha iyi olduğunu göstermiştir. Klasik kümelemede olayların ait oldukları kümeyi 1 üyelik değeri ile temsil etmesi, önemliliği yüksek olayları içeren test dizilerinin önceliğini bulanık kümelerde yer alan olaylardan daha fazla artırmaktadır.

- BCO ile kümeleme için MATLAB'ın bulanık mantık toolbox'ında bulunan hazır fonksiyon kullanılış ve bulanıklık indeksi 2 olarak alınmıştır. Fonksiyondaki parametreler değiştirilerek kümeleme performansı dolayısıyla test dizilerinin hata yakalama başarımının artırılabilceği öngörülmektedir. İlerleyen çalışmalarda farklı parametrelerle denemeler yapılacaktır.
- Önerilen önceliklendirme metotları önceki test süreçlerinde ortaya çıkarılan hatalar dikkate alınacak şekilde geliştirilebilir.
- Bu çalışmada olayları temsil etmek için önerilen faktörlerin eşit ağırlığa sahip olduğu varsayılmıştır. Gelecek çalışmalarda temel bileşenler analizi, faktör analizi gibi çok değişkenli istatistiksel metotlar ile faktörler ağırlıklandırılarak önceliklendirmenin hata yakalama başarımına etkileri araştırılacaktır.
- Yöntem olay çiftlerini yani 2'lileri kapsayan test dizilerine uygulanmıştır. 3'lü, 4'lü, … n'li olay gruplarının hata ortaya çıkma başarıları da test için önemli bir konudur. Bu nedenle önerilen yöntemin gelecek çalışmalarda 3'lü, 4'lü … n'li olay grupları içinde genişletilmesi planlanmaktadır.
- Önerilen test önceliklendirme yöntemleri ÇPP esasına göre üretilen test takımına uygulanmıştır. Bu test takımında eşit uzunluğa sahip test dizileri olduğu gibi bazı test dizileri arasında uzunluk farkı da çok fazladır. Gelecek çalışmalarda kapsam kriteri devredışı bırakılarak eşit uzunlukta test dizileri üretilmesi kısıtı ile üretilen test takımlarına önerilen önceliklendirme yöntemleri uygulanabilir.

KAYNAKLAR

Acharya, A.A., Mohapatra, D.P. ve Panda, N. (2010) Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UML Sequence Diagram. *Int J of Adv Comput Sci and Appl* 1(6):108-113.

Ahalt, S.C., Krishnamurty, P., Chen, P. ve Melton, D.E. (1990) Competitive Learning Algorithms for Vector Quantization. *J Neural Netw* 3: 277-291.

Aldenderfer, M.S. ve Blashfield, R.K. (1984) *Cluster Analysis.* Sage Publications, Los Angeles, 88s.

Anderberg, M.R. (1973) *Cluster Analysis for Applications*. Academic Press, New York, 359s.

Anonim (2002) The economic impacts of inadequate infrastructure for software testing. http://www.nist.gov/director/planning/upload/report02-3.pdf.

Armour, P.G. (2003) *Laws of Software Process: A New Model for the Production and Management of Software*. Auerbach Publications, UK, 272s.

Athira, B. ve Samuel, P. (2010) Web Services Regression Test Case Prioritization. *International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, October 8-10, Krackow, 438- 443.

Ball, T. (1998) On the limit of Control Flow Analysis for Regression Test Selection. *ACM SIGSOFT International Symposium On Software Testing And Analysis (ISSTA'98),* March 2-4, Clearwater Beach, Florida, 134-142.

Beizer, B. (1990) *Software Testing Techniques*, Van Nostrand Reinhold, New York, NY, 2. Baskı 580s.

Belli, F. (2001) Finite-State Testing and analysis of Graphical User Interfaces. *Proc 12th International Symposium on Software Reliability Engineering (ISSRE'01)*, November 27-30, Hong Kong, China, 34- 43.

Belli, F. (2008) Lecture Notes, http://adt.uni-paderborn.de/uploads/media/Class1.pdf

Belli, F. ve Budnik, C.J. (2007) Test Minimization for Human-Computer Interaction. *J Appl Intell* 7(2): 161-174.

Belli, F. ve Gökçe, N. (2010) Test Prioritization at Different Modeling Levels. *Communications in Computer and Information Science (ASEA 2010)*, December 13-15, Jeju, Korea, 117: 130-140.

Belli, F., Budnik, Ch.J. ve White, L. (2006) Event-Based Modeling, Anaylsis and Testing of User Interactions-Approach and Case Study. *J Softw Test Verif Reliab* 16(1): 3-32.

Belli, F., Eminov, M. ve Gökçe, N. (2007a) Prioritizing Coverage-Oriented Testing Process- An Adaptive-Learning-Based Approach and Case Study. *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, July 23-27, Beijing, China, 2: 197-203.

Belli, F., Eminov, M. ve Gökçe, N. (2007b) Coverage-Oriented, Prioritized Testing – A Fuzzy Clustering Approach and Case Study. *Third Latin-American Symposium on Dependable Computing (LADC 2007)*, September 26-28, Morelia, Mexico, LNCS, 4746: 95-110.

Belli, F., Eminov, M. ve Gökçe, N. (2009) Model-Based Test Prioritization- a Comparative Soft- Computing Approach and Case Studies. *KI 2009: Advances in Artificial Intelligence*, September 15-18, Paderborn, Germany, LNCS, 5803: 427-434.

Belli, F., Eminov, M., Gökçe, N. ve Wong, E. (2011) Prioritizing Coverage-Oriented Testing Process — An Adaptive-Learning-Based Approach and Case Study.20:1-21. Wong W.E., Cukic B. *Series on Software Engineering and Knowledge Engineering, Adaptive Control Approach For Software Quality Improvement*, World Scientific Publishing, Singapore, 299s.

Belli, F., Güler, N., ve Linschulte, M. (2010) Are longer test sequences always better? - A Reliability Theoretical Analysis. *4th International Conference on Secure Software Integration and Reliability Improvement (SSIRI 2010)*, June 9-11, Singapore, IEEE Computer Society, 78-85.

Bezdek, J.C. (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press. New York, 272s.

Binder, R.V. (2000) *Testing Object-Oriented Systems: models, patterns, and tools*, Addison Wesley Professional, Boston, 1248s.

Bonissone, P.P. (1997) Soft Computing: the Convergence of Emerging Reasoning Technologies, *Soft Comput – A Fusion of Found, Methodol and Appl* 1(1): 6-18.

Broy, M., Jonsson, B., Katoen, J.P., Leucker, M. ve Pretschner, A. (2005) *Model-Based Testing of Reactive Systems: Advanced Lectures*, Springer, Berlin, 667s.

Bryce, C. ve Memon, A.M. (2007) Test Suite Prioritization by Interacting Coverage, *In Proceedings of Workshop on Domain - Specific Approaches to Software Test Automation (DoSTA2007)*, September 3-8, Dubrovnik, Croatia, 1-7.

Bryce, R.C., Sampath, S. ve Memon, A.M. (2011) Developing a Single Model and Test Prioritization Stategies for Event-Driven Software, *IEEE Trans on Softw Eng* 37(1): 1-18.

Budnik C. J. (2006) *Test Generation Using Event Sequence Graphs*, PhD Thesis, Paderborn University, Paderborn, 166s.

Carroll, J. ve Long, D. (1989) *Theory of Finite Automata with an Introduction to Formal Languages*. Prentice Hall, Englewood Cliffs, 432s.

Chen, J., Subramaniam, S. 2002. Specification-based Testing for GUI-based Applications. *J Softw Qual* 10(3): 205-224.

Chen, Y., F., Rosenblum, D. S. ve Vo, K. P. (1994) Test Tube: A system for Selective Regression Testing. *16th International Conference on Software Engineering (ICSE-16)*, May 16-21, Sorrento, Italy, 211-222.

Cheng, K.T. ve Krishnakumar, A.S. (1993) Automatic functional test generation using the extended finite state machine model. *The 30th International Design Automation Conference (DAC '93)*, Juni 14-18, Dallas, USA, 86-91.

Chou, C., Ho, B. ve Sheu, J.T. (1995) Material characterization by ultrasonics using unsupervised competitive learning. *J Pattern Recognit Lett*, 16(7): 769-777.

Chow, T. (1978) Testing software design modeled by finite state machines. *IEEE Trans on Softw Eng*, 4(3): 178-187.

Cook, S.A. (1971) The complexity of theorem proving procedures. *The third annual ACM Symposium on Theory of Computing (STOC '71)*, May 3-5, Shaker Heights, Ohio, 151–158.

Craig, R.D. ve Jaskiel,S. P. (2002) *Systematic Software Testing*, Artech House Publishers, Bostoni, London, 536s.

De Jong, K.A. ve Spears, W.M. (1989) Using Genetic Algorithms to Solve NP-Complete Problems. *the Third International Conference on Genetic Algorithms (ICGA'89)*, Juni 4-7, George Mason University, Virginia, 124-132.

Dick, J. ve Faivre, A. (1993) Automating the generation and sequencing of test cases from model-based specifications. *the First International Symposium of Formal Methods (FME'93)*, April 19-23, Odense, Denmark, LNCS, 670: 268-284.

Dssouli, R., Saleh, K., Aboulhamid, E., En-Nouaary, A. ve Bourhfir, C. (1999) *Test* development for communication protocols: towards automation. *Comput Netw* 31(17): 1835-1872.

Dunn, J. C. (1973) A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *J of Cybern* 3(3): 32-57.

Edmonds, J. ve Johnson, E.L. (1973) Matching: Euler Tours and the Chinese Postman, *Math Program* 5(1): 88-124.

Edvardson, J. (1999) A survey on automatic test data generation. *The 2nd Conference on Computer Science and Engineering in Linköping (CCSSE'99)*, October 21-21, Norrköping, Sweden, 21-28.

Elbaum, S., Malishevsky, A. ve Rothermel, G. (2001) Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization, *23th international Conference on Software Engineering, (ICSE-01)*, May 12-19, Toronto, Canada, 329-338.

Elbaum, S., Malishevsky, A. ve Rothermel, G. (2001) Prioritizing test case for regression testing. *the International Symposium on Software Testing and Analysis*, August 21-24, Protland, OR, USA, 102-112.

Elbaum, S., Rothermel, G., Kanduri, S. ve Malishevsky, A. (2004) Selecting A Cost Effective Test Case Prioritization Technique. *J Softw Qual Control* 12(3): 185-210.

Eminov, M. (2003) Fuzzy c-Means Based Adaptive Neural Network Clustering. *Int 12th Turkish Symposium Artificial Intelligence and Neural Networks (TAINN 2003)*, 2-4 Temmuz, Çanakkale, Turkey, E2:154-162.

Eminov, M. ve Gokce, N. (2005) Neural Network Clustering Using Competitive Learning Algorithm. *Int 14th Turkish Symposium Artificial Intelligence and Neural Networks (TAINN 2005)*, 16-17 Haziran, İzmir, Turkey, 161-168.

Ensan, A., Bagheri, E., Asadi, M., Gasevic, D. ve Biletskiy, Y. (2011) Goal-Oriened Test Case Selection and Prioritization for Product Line Feature Models. *8th International Conference on Information Technology: New Generations*, April 11-13, Nevada, USA, 291-298.

Fraser, G. ve Wotawa, F. (2007) Redundancy Based Test-Suite Reduction. *Fundamental Approaches to Software Engineering (FASE'07),* 24 March-1 April, Braga, Protugal, LNCS, 4422: 291-305.

Fraser, G. ve Wotawa, F. (2007) Test Case Prioritization with Model Checkers. *the 25th conference on IASTED International Multi-Conference: Software Engineering (SE'07)*, February 13-15, Innsbruck, Austria, 267-272.

Fu, L.M. (1994) *Neural Networks in Computer Intelligence*, McGraw-Hill, New York, 460s.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Profesional, Boston, 432s.

Goodenough, J.B. ve Gerhart, S. (1975) Toward a Theory of Test Data Selection. *IEEE Trans On Softw Eng* 1(2): 156-173.

Gökçe, N., Eminov, M. ve Belli, F. (2006) Coverage-Based, Prioritized testing Using Neural Network Clustering, *The 21st International Symposium on Computer and Information Sciences (ISCIS 2006)*, November 1-3, İstanbul, Turkey, LNCS, 4263: 1060-1071.

Grossberg, S. (1987) Competitive Learning: from interactive activation to adaptive resonance, *Cogn Sci* 11(1): 23-63.

Halkidi, M., Batistakis, Y. ve Vazirgiannis, M. (2001) On Clustering Validation Techniques. *J of Intell Inf Syst* 7: 107-145.

Hamlet, D. (1994) Foundation of Software Testing: Dependability Theory. *The 2nd ACM SIGSOFT symposium on Foundations of software engineering (SIGSOFT'94)*, December 6-9, New Orleans, USA, 19(5): 128-139.

Han, J. ve Kamber, M. (2006) *Data Mining Concept and Techniques*. Morgan Kaufmann Publisher, Massachusetts, 800s.

Harel, D. (1987) Statecharts: A Visual Formalism for Complex Systems. *Sci of Comput Program* 8(3): 231-274.

Harrold, M.J., Gupta, R. ve Soffa, M.L. (1993) Methodology for Controlling the Size of a Test Suite. *ACM Trans Softw Eng and Methodol* 2(3): 270-285.

Haykin, S. (1994) *Neural Networks A Comprehensive Foundation*, Prentice Hall, Macmillan, 842s.

Hebb, D.O. (1949) *The Organization of Behaviour: A Neouropsychological Theory*. Psychology Press, New York, 378s.

Heimdahl, M.P.E. ve George, D. (2004) Test-suite reduction for model based tests: effects on test quality and implications for testing. *19th International*

Conference on Automated Software Engineering, September 20-25, Linz, Austria, 176-185.

Heimdahl, M.P.E., George, D. ve Weber, R. (2004) Specification Test Coverage Adequacy Criteria = Specification Test Generation Inadequacy Criteria?, *8th IEEE International Symposium on High Assurance Systems Engineering (HASE'04)*, March 25-26, Tampa, FL, USA, 178-186.

Holland, J.H. (1975) *Adaption in Natural and Artificial Systems*, University of Michigan Pres, Ann Arbor, MI,183s.

Hoppner, F., Klawonn, F., Kruse, R. ve Runkler, T. (1999) *Fuzzy Cluster Analysis*, John Wiley, New York, 300s.

Jain, A.K., Murty, M.N. ve Flyn, P.J. (1999) Data Clustering: A Review, *ACM Comput Surv* 31(3): 264-323.

Jang, S.R., Sun, C.T. ve Mizutani, E. (1997) *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall Inc, USA, 614s.

Jeffrey, D. ve Gupta, N. (2006) Test Case Prioritization Using Relevant Slices. *The 30th Annual International Computer Software and Applications Conference*, September 17-21, Chicago, USA, 1: 411-420.

Jiang, B., Chan, W.K. ve Tse, T.H. (2011) On Practical Adequate Test Suite for Integrated Test Case Prioritization and Fault Localizaion. *The 11th International Conference an Quality Software (QSIC 2011),* July 13-14, Madrid, Spain, 1-11.

Jourdan, G.V., Ritthiruangdech, P. ve Ural, H. (2006) Test Suite Reduction Based on Dependence Analysis. *The 21st international conference on Computer and Information Sciences (ISCIS 2006),* November 1-3, İstanbul, Turkey, LNCS 4263: 1021-1030.

Kim, D.J., Park , Y.W. ve Park, D.J. (2001) A Novel Validity Index for Determination of the Optimal Number of Clusters. *IEICE Trans Inf & Syst* E84-D,(2): 281-285.

Kim, J.M. ve Porter, A. (2002) A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments. *The 24th International Conference on Software Engineering (ICSE'02),* May 19-25, Orlando, Florida, USA, 119-129.

Kohonen, T. (1970) Correlation matrix memories. *IEEE Trans Comput* C-21: 353-359.

Kohonen, T. (1982) Self-organized formation of topologically correct feature maps. *Biol Cybern* 43: 59-69.

Kohonen, T. (1984) *Self-Organization and Associative Memory*, Springer, Berlin, 255s.

Kohonen, T. (1989) *Self-organization and Associative Memory*. Springer-Verlag, Berlin, 312s.

Korel, B., Tahat, L. H. ve Harman, M. (2005) Test Prioritization Using system Models. *The 21st IEEE International Conference on software Maintenance (ICCM'05),* September 25-30, Budapest, Hungary, 559-568.

Krishnamoorthi, R. ve Sahaaya Arul Mary, S.A. (2009) Requirement Based System Test Case Prioritization of New and Regression Test Cases. *Int J of Softw Eng and Knowl Eng* 19(3): 453-475.

Ledru, Y., Petrenko, A. ve Boroday, S. (2009) Using String Distances for Test Case Prioritisation, *24th IEEE/ACM International Conference on Automated Software Engineering (ASE2009),* November 16-20, Aucland, New Zealand, 510-514.

Ma, Z. ve Zhao, J. (2008) Test Case Prioritization Based on Analysis of Program Structure. *15th Asia-Pacific Software Engineering Conference (APSEC'08),* December 3-5, Beijing, China, 471-478.

Maeda, M. ve Miyajima, H. (2000) Competitive Learning Algorithm Founded on Adaptivity and Sensitivity Deletion Method. *IEICE Trans Fundam Commun Comput Sci* E83-A(12):2770-2774.

Maeda, M., Miyajima, H. ve Marashima, S. (1996) An Adaptive Learning and Self-deleting Neural Network for Vector Quantization. *IEICE Trans Fundamof Electron Commun Comput Sci* E79-A(11): 1886-1893.

Malaiya, Y.K. (1995) Antirandom testing: Getting the most out of black-box testing. *Sixth International Symposium on Software Reliability Engineering, (ISSRE'95),* October 24-27, Toulouse, France, 86-95.

Martinez, T.M., Berkovich, S.G. ve Schulten, K.J. (1993) "Neural Gas" Network for Vector Quantization and Its Application to Times-series Prediction. *IEEE Trans Neural Netw* 4(4): 558-569.

Masri, W., Podgurski, A. ve Leon, D. (2007) An Empirical Study of Test case Filtering Techniques Based on Exercising Information Flows. *IEEE Trans on Softw Eng* 33(7): 454-477.

Mathur, A. P. (2007) *Foundations of Software Testing*, Addison-Wesley Professional, Boston, 689s.

McCulloch, W.S. ve Pitts, W.H. (1943) A logical calculus of the ideas immanent in nervous activity. *Bull of Mat Biophys,* 5: 115-133.

Melin, P. ve Castillo, O. (2005) *Studies in Fuzziness and Soft Computing*. Springer-Verlag, Berlin, 272s.

Memon, A.M., Pollack, M.E. ve Soffa, M.L. (2000) Automated Test Oracles for GUIs. *ACM SIGSOFT Software Engineering Notes*, November 6-10, San Diego, California, USA, 25(6): 30-39.

Mohanty, S., Acharya, A.A. ve Mohapatra, D.P. (2011) A Survey on Model Based Test Case Prioritization. *Int J of Comput Sci and Inf Technol (IJCSIT)* 2(3): 1042-1047.

Myers, G. (1979) *The Art of Software Testing*, John Wiley and Sons, New York, 192s.

Offutt, J. ve Abdurazik, A. (1999) Generating Tests from UML specifications. The *2nd International Conference on the Unified Modeling Language (UML99)*, October 28-30, Fort Collins, CO, USA, 416-429.

Özdamar, K. (2002) *Paket Programlar ile İstatistiksel Veri Analizi-2*, Kaan Kitabevi, Eskişehir, 513s.

Qu, B., Nie, C., Xu, B. ve Zhang, X. (2007) Test Case Prioritization for Black Box Testing. *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, July 23-27, Beijing, China, 1: 465-472.

Qu, X., Cohen, M.B. ve Woolf, K.M. (2007) Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization. *The 23th IEEE International Conference on Software Maintenance (ICSM2007)*, October 2-5, Paris, France, 255-264.

Rothermel, G. ve Harrold, M. J. (1993) A safe, efficient algorithm for regression test selection. *The Conference on Software Maintance*, September 27-30, Montreal, Canada, 358-367.

Rothermel, G. ve Harrold, M.J. (1997) A safe,efficient regression test selection technique, *ACM Transactions on Software Engineering and Methodology*, 6(2): 173-210.

Rothermel, G., Harrold, M.J., Ronne, J. Von, ve Hong, C. (2002) Empirical Studies of Test-Suite Reduction. *J Softw Test Verif and Reliab* 12(4): 219-249.

Rummelhart D.E. ve Zipser, D. (1985) Competitive Learning, *J Cogn Sci*, 9: 75-112.

Salem, Y.I. ve Hassan, R. (2010) Requirement-based Test Case Generation and Prioritization. *International Computer Engineering Conference (ICENCO 2010)*, December 27-28, Giza, Egypt, 152-157.

Srikanth, H., Williams, L. ve Osborne, J. (2005) System Test Case Prioritization of New and Regression Tests. *In International Symposium of Empirical Software Engineering (ISESE 2005)*, November 17-18, Noosa Heads, Australia, 1-10.

Srivastava, A. ve Thiagrajan, J. (2002) Effectively Prioritizing Test in Development Environment. *The ACM International Symposium on Software Testing and Analysis (ISSTA2002)*, July 22-24, Roma, Italy, 97-106.

Tan, P.N., Steinbach, M. ve Kumar, V. (2006) *Introduction to Data Mining*, Addison-Wesley, Boston, 769s.

Tatlıdil, H. (2002) *Uygulamalı Çok Değişkenli İstatistiksel Analiz*, Akademi Matbaası, Ankara, 424s.

URL, ISELTA, http://www.iselta.de/

Walcott, K.R., Soffa, M.L., Kapfhammer, G.M. ve Roos, R.S. (2006) Time-Aware Test Suite Prioritization. *The ACM International Symposium on Software Testing and Analysis (ISSTA2006)*, July 17-20, Portland, Maine, USA, 1-12.

Wiener, N. (1948) *Cybernetics: Control and Communication in the Animal and the machine*, John Wiley & Sons, New York, 194s.

Willshaw, D.J. ve Malsburg, Ch.V.D. (1976) How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London,* November 12, Series B, 194(1117): 431-445.

Wong, W., Horgan, J., London S. ve Agrawal, H. (1997) A study of effective regression testing in practice. *8th International Symposium on Software Reliability Engineering, (ISSRE'97),* November 2-5, Albuquerque, New Mexico, 230-238.

Wong, W.E., Horgan, J.R., London, S. ve Mathur, A.P. (1995) Effect of Test Set Minimization on Fault Detection Effectiveness. *The 17th international conference on Software engineering, (ICSE'95*), April 23-30, Seattle, Washington, USA, 41-50.

Xu, L., Krzyzak, A. ve Oja, E. (1993) Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection. *IEEE Trans on Neural Netw* 4(4): 636-649.

Yannakakis, M. ve Lee. D. (1995) Testing finite state machines: Fault detection. *J of Comput and Syst Sci* 50: 209-227.

Zadeh, L.A. (1965) Fuzzy Sets. *Inf and Control* 8(3): 338–353.

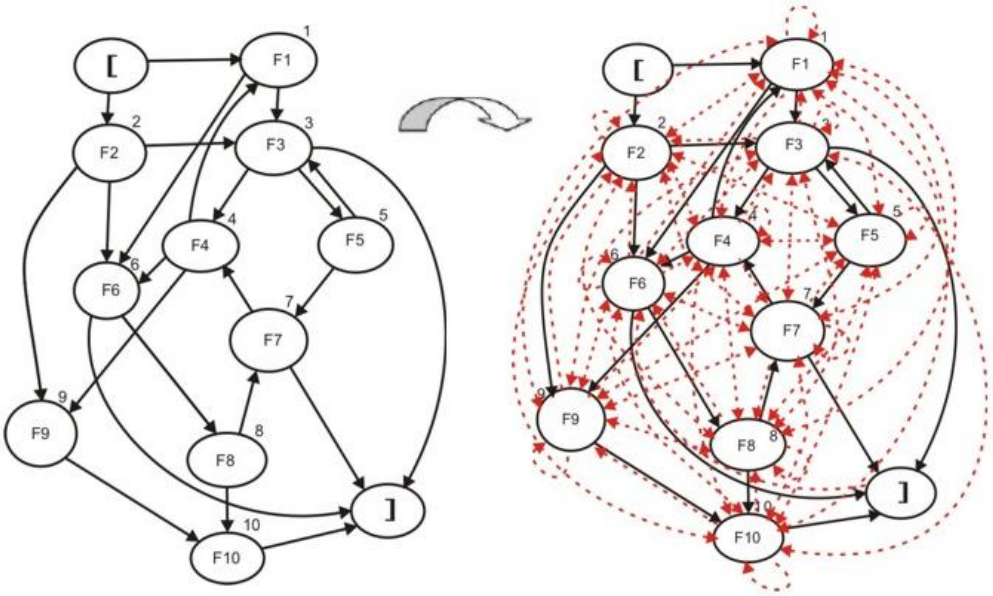
Zadeh, L.A. (1994) Fuzzy Logic. *Neural Netwo, and Soft Comput, Commun of ACM,* 37(3): 77-84.

Zadeh, L.A. (1996) *Fuzzy Sets, Fuzzy Logic, Fuzzy Systems*, World Scientific Press, USA, 826s.

Zhu, H., Hall, P.A.V. ve May, J.H.R. (1997) Software unit test coverage and adequacy. *ACM Comput Surv*, 29(4): 365-427.

EKLER

Ek 2.1. Örnek ODG'ler için test önceliklendirme uygulamaları



Şekil E2. 1. ODG_2

Tablo E2. 1. ODG_2 Veri Seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	1	4	9	3	0,399	9	8	3	0,227
	x_2	1	4	10	2	0,054	8	1	4	0,333
	x_3	2	6	9	3	0,399	8	9	4	0,341
	x_4	3	5	10	5	0,242	8	7	3	0,162
	x_5	3	3	9	0	0,242	9	10	2	0,171
	x_6	2	5	9	1	0,242	9	4	4	0,246
	x_7	4	4	10	6	0,399	9	11	3	0,162
	x_8	3	3	9	0	0,242	9	5	3	0,161
	x_9	2	3	2	0	0,242	10	6	2	0,222
	x_{10}	3	3	1	2	0,242	10	7	3	0,242

Tablo E2. 2. ODG_2 için standartlaştırılmış veri seti

		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}	Gözlem ort.
Standartlaştırılmış gözlem değerleri	x_1	0,14	0,36	0,00	1,21	0,00	0,38	-0,14	0,40	-1,45	**0,10**
	x_2	-1,22	0,65	1,58	-2,04	0,00	-0,10	1,22	-1,95	-1,45	**-0,37**
	x_3	-1,22	0,36	1,70	1,21	1,90	0,38	1,22	0,74	-0,41	**0,65**
	x_4	-1,22	0,65	-0,96	-0,27	0,95	1,33	-0,14	0,07	0,62	**0,12**
	x_5	0,14	0,36	-0,83	-0,27	-0,95	-1,05	-1,49	1,08	0,62	**-0,27**
	x_6	0,14	0,36	0,29	-0,27	0,95	-0,57	1,22	-0,94	-0,41	**0,08**
	x_7	0,14	0,65	-0,96	1,21	0,00	1,81	-0,14	1,41	1,66	**0,64**
	x_8	0,14	0,36	-0,98	-0,27	-0,95	-1,05	-0,14	-0,61	0,62	**-0,32**
	x_9	1,49	-1,73	-0,07	-0,27	-0,95	-1,05	-1,49	-0,27	-0,41	**-0,53**
	x_{10}	1,49	-2,02	0,23	-0,27	-0,95	-0,10	-0,14	0,07	0,62	**-0,12**

Tablo E2. 3. ODG_2 için GO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 6 8 7 4 1 3 5 7]	41	2
TOD_2	[2 6 8 7 4 9 10]	43	1
TOD_3	[2 3 5 3]	18	5
TOD_4	[2 6 8 10]	28	3
TOD_5	[2 9 10]	25	4
TOD_6	[1 6]	9	6

Tablo E2. 4. ODG_2 için YO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 6 8 7 4 1 3 5 7]	28	1
TOD_2	[2 6 8 7 4 9 10]	22	2
TOD_3	[2 3 5 3]	10	4
TOD_4	[2 6 8 10]	11	3
TOD_5	[2 9 10]	9	5
TOD_6	[1 6]	3	6

Tablo E2. 5. ODG_2 için BCO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 3 4 6 8 7 4 1 3 5 7]	20,0072	1
TOD_2	[2 6 8 7 4 9 10]	16,8923	2
TOD_3	[2 3 5 3]	8,8131	5
TOD_4	[2 6 8 10]	9,2366	4
TOD_5	[2 9 10]	10,4457	3
TOD_6	[1 6]	2,1831	6

Tablo E2. 6. ODG_2 için karşılaştırmalı sonuçlar

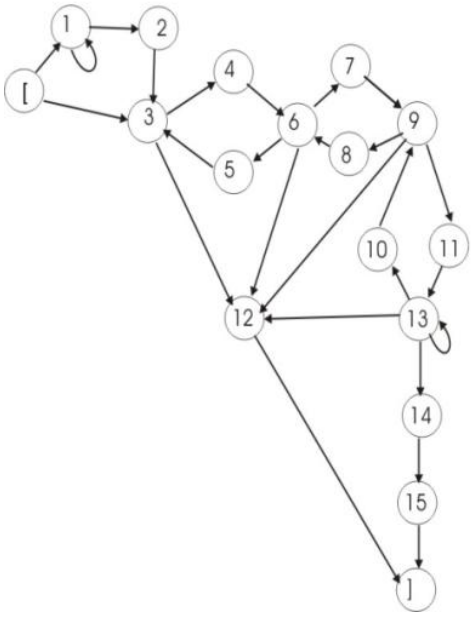
No	Test Dizisi	Öncelik Sıraları		
		GO	YÖ	BCO
TOD_1	[1 3 4 6 8 7 4 1 3 5 7]	2	1	1
TOD_2	[2 6 8 7 4 9 10]	1	2	2
TOD_3	[2 3 5 3]	5	4	5
TOD_4	[2 6 8 10]	3	3	4
TOD_5	[2 9 10]	4	5	3
TOD_6	[1 6]	6	6	6

Tablo E2. 7. ODG_2 için Friedman testi sonuçları

	Test İstatistikleri
Ki-Kare	10,333
Serbestlik Derecesi	2
Anlamlılık Değeri	0,006

Tablo E2. 8. ODG_2 için Kendall Tau testi sonuçları

			YÖ	GO	BCO
Kendall's Tau_b	YÖ	Korelasyon katsayısı	1,000	0,552	0,966
		Anlamlılık (çift-taraflı)	0,000	0,126	0,007
	GO	Korelasyon katsayısı		1,000	0,467
		Anlamlılık (çift-taraflı)		0,000	0,188
	BCO	Korelasyon katsayısı			1,000
		Anlamlılık (çift-taraflı)			0,000



Şekil E2. 2. ODG_3

Tablo E2. 9. ODG_3 için veri seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	1	4	16	18	0,3989	13	2	2	0,2222
	x_2	2	2	14	10	0,3989	14	3	1	0,1111
	x_3	1	5	14	8	0,242	13	8	6	0,1348
	x_4	2	2	14	12	0,3989	14	5	5	0,1126
	x_5	4	2	14	3	0,3989	14	7	1	0,1111
	x_6	3	5	14	2	0,242	12	7	6	0,1376
	x_7	4	2	14	9	0,3989	14	4	5	0,113
	x_8	6	2	14	7	0,3989	14	6	1	0,125
	x_9	5	5	14	5	0,242	13	10	4	0,1357
	x_{10}	8	2	14	10	0,3989	14	9	1	0,0909
	x_{11}	6	2	14	6	0,3989	14	6	3	0,109
	x_{12}	2	5	1	5	0,0044	14	11	4	0,113
	x_{13}	7	6	14	3	0,3989	11	7	4	0,1393
	x_{14}	8	2	2	2	0,3989	14	8	1	0,1111
	x_{15}	7	2	1	0	0,3989	14	9	1	0,1111

Tablo E2. 10. ODG_3 standartlaştırılmış veri seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	-1,3592	0,5104	0,8236	2,4189	0,5093	-0,5098	-1,8890	-0,5092	3,2275
	x_2	-0,9595	-0,7655	0,4492	0,7114	0,5093	0,5826	-1,4954	-1,0184	-0,4679
	x_3	-1,3592	1,1483	0,4492	0,2846	-0,8764	-0,5098	0,4722	1,5275	0,3204
	x_4	-0,9595	-0,7655	0,4492	1,1383	0,5093	0,5826	-0,7084	1,0184	-0,4180
	x_5	-0,1599	-0,7655	0,4492	-0,7826	0,5093	0,5826	0,0787	-1,0184	-0,4679
	x_6	-0,5597	1,1483	0,4492	-0,9960	-0,8764	-1,6021	0,0787	1,5275	0,4135
	x_7	-0,1599	-0,7655	0,4492	0,4980	0,5093	0,5826	-1,1019	1,0184	-0,4047
	x_8	0,6396	-0,7655	0,4492	0,0711	0,5093	0,5826	-0,3148	-1,0184	-0,0055
	x_9	0,2399	1,1483	0,4492	-0,3557	-0,8764	-0,5098	1,2593	0,5092	0,3504
	x_{10}	1,4392	-0,7655	0,4492	0,7114	0,5093	0,5826	0,8658	-1,0184	-1,1398
	x_{11}	0,6396	-0,7655	0,4492	-0,1423	0,5093	0,5826	-0,3148	0,0000	-0,5377
	x_{12}	-0,9595	1,1483	-1,9841	-0,3557	-2,9735	0,5826	1,6528	0,5092	-0,4047
	x_{13}	1,0394	1,7863	0,4492	-0,7826	0,5093	-2,6944	0,0787	0,5092	0,4701
	x_{14}	1,4392	-0,7655	-1,7969	-0,9960	0,5093	0,5826	0,4722	-1,0184	-0,4679
	x_{15}	1,0394	-0,7655	-1,9841	-1,4229	0,5093	0,5826	0,8658	-1,0184	-0,4679

Tablo E2. 11. ODG_3 için GO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Derecesi
TOD_1	[1 1 2 3 4 6 5 3 12]	65	3
TOD_2	[3 4 6 7 9 8 6 12]	63	4
TOD_3	[3 4 6 7 9 11 13 13 10 9 12]	67	2
TOD_4	[3 4 6 7 9 11 13 12]	57	5
TOD_5	[3 4 6 7 9 11 13 14 15]	68	1

Tablo E2. 12. ODG_3 için YÖ tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Derecesi
TOD_1	[1 1 2 3 4 6 5 3 12]	31	2
TOD_2	[3 4 6 7 9 8 6 12]	27	5
TOD_3	[3 4 6 7 9 11 13 13 10 9 12]	36	1
TOD_4	[3 4 6 7 9 11 13 12]	28	4
TOD_5	[3 4 6 7 9 11 13 14 15]	29	3

Tablo E2. 13. ODG_3 için BCO tabanlı dayalı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Derecesi
TOD_1	[1 1 2 3 4 6 5 3 12]	17,4582	3
TOD_2	[3 4 6 7 9 8 6 12]	15,7425	4
TOD_3	[3 4 6 7 9 11 13 13 10 9 12]	20,7354	2
TOD_4	[3 4 6 7 9 11 13 12]	15,5019	5
TOD_5	[3 4 6 7 9 11 13 14 15]	20,9699	1

Tablo E2. 14. ODG_3 için karşılaştırmalı sonuçlar

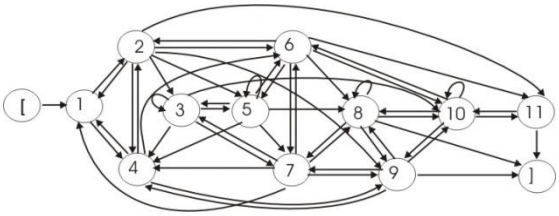
No	Test Dizisi	Öncelik Sıraları		
		GO	***YO***	***BCO***
TOD_1	[1 1 2 3 4 6 5 3 12]	3	2	3
TOD_2	[3 4 6 7 9 8 6 12]	4	5	4
TOD_3	[3 4 6 7 9 11 13 13 10 9 12]	2	1	2
TOD_4	[3 4 6 7 9 11 13 12]	5	4	5
TOD_5	[3 4 6 7 9 11 13 14 15]	1	3	1

Tablo E2. 15. ODG_3 için Friedman testi sonuçları

	Test İstatistikleri
Ki-Kare	10,000
Serbestlik Derecesi	2
Anlamlılık Değeri	0,007

Tablo E2. 16. ODG_3 için Kendall Tau testi sonuçları

			YÖ	**GO**	**BCO**
Kendall's Tau_b	YÖ	Korelasyon katsayısı	1,000	0,316	0,949
		Anlamlılık (çift-taraflı)	0,000	0,448	0,023
	GO	Korelasyon katsayısı		1,000	0,400
		Anlamlılık (çift-taraflı)		0,000	0,327
	BCO	Korelasyon katsayısı			1,000
		Anlamlılık (çift-taraflı)			0,000



Şekil E2. 3. ODG_4

Tablo E2. 17. ODG_4 veri seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	1	6	12	15	0,0540	9	10	7	0,1303
	x_2	2	10	12	3	0,0001	4	13	9	0,163
	x_3	3	9	12	2	0,2420	6	7	6	0,0899
	x_4	2	10	12	9	0,0540	7	15	7	0,1314
	x_5	3	10	12	18	0,0540	5	17	6	0,1155
	x_6	3	11	12	12	0,2420	5	26	7	0,1279
	x_7	4	11	12	4	0,2420	5	23	7	0,1217
	x_8	4	11	12	2	0,2420	6	27	6	0,1164
	x_9	3	10	12	3	0,3989	6	19	6	0,1852
	x_{10}	4	11	12	2	0,2420	6	25	6	0,1481
	x_{11}	3	5	12	1	0,2420	9	21	2	0,056

Tablo E2. 18. ODG_4 standartlaştırılmış veri seti

		Faktörler								
		x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}	x_{i9}
Olaylar	x_1	-2,0226	-1,6712		1,4189	-1,0502	1,7601	-1,2649	0,4332	0,1266
	x_2	-0,9631	0,2639		-0,5736	-1,4887	-1,3627	-0,8161	1,6246	1,0770
	x_3	0,0963	-0,2199		-0,7396	0,4802	-0,1136	-1,7138	-0,1625	-1,0477
	x_4	-0,9631	0,2639		0,4226	-1,0502	0,5110	-0,5169	0,4332	0,1585
	x_5	0,0963	0,2639		1,9170	-1,0502	-0,7381	-0,2176	-0,1625	-0,3036
	x_6	0,0963	0,7477		0,9208	0,4802	-0,7381	1,1289	0,4332	0,0568
	x_7	1,1558	0,7477		-0,4076	0,4802	-0,7381	0,6801	0,4332	-0,1234
	x_8	1,1558	0,7477		-0,7396	0,4802	-0,1136	1,2785	-0,1625	-0,2774
	x_9	0,0963	0,2639		-0,5736	1,7582	-0,1136	0,0816	-0,1625	1,7222
	x_{10}	1,1558	0,7477		-0,7396	0,4802	-0,1136	0,9793	-0,1625	0,6439
	x_{11}	0,0963	-2,1550		-0,9057	0,4802	1,7601	0,3808	-2,5452	-2,0330

Tablo E2. 19. ODG_4 için GO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik sırası
TOD_1	[1 2 3 3 5 3 7 8 8 9 7 9 4 9 10 10 11 10 9 8 10 8 7 6 10 6 8]	131	1
TOD_2	[1 4 2 6 5 5 6 2 4 1 2 11]	81	2
TOD_3	[1 2 1 4 6 7 3 4 2 5 7 4 2 5 4 6 5 8 7 6 11]	131	1
TOD_4	[1 2 3 10 9 7 1 2 9]	56	3

Tablo E2. 20. ODG_4 için YÖ tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 2 3 3 5 3 7 8 8 9 7 9 4 9 10 10 11 10 9 8 10 8 7 6 10 6 8]	54	2
TOD_2	[1 4 2 6 5 5 6 2 4 1 2 11]	49	3
TOD_3	[1 2 1 4 6 7 3 4 2 5 7 4 2 5 4 6 5 8 7 6 11]	69	1
TOD_4	[1 2 3 10 9 7 1 2 9]	30	4

Tablo E2. 21. ODG_4 için BCO tabanlı test önceliklendirme sonuçları

No	Test Dizisi	PrefD(TODq)	Öncelik Sırası
TOD_1	[1 2 3 3 5 3 7 8 8 9 7 9 4 9 10 10 11 10 9 8 10 8 7 6 10 6 8]	77,9261	1
TOD_2	[1 4 2 6 5 5 6 2 4 1 2 11]	29,1535	4
TOD_3	[1 2 1 4 6 7 3 4 2 5 7 4 2 5 4 6 5 8 7 6 11]	46,6925	2
TOD_4	[1 2 3 10 9 7 1 2 9]	29,3307	3

Tablo E2. 22. ODG_3 için karşılaştırmalı sonuçlar

No	Test Dizisi	Öncelik Sıraları		
		GO	YÖ	BCO
TOD_1	[1 2 3 3 5 3 7 8 8 9 7 9 4 9 10 10 11 10 9 8 10 8 7 6 10 6 8]	1	2	1
TOD_2	[1 4 2 6 5 5 6 2 4 1 2 11]	2	3	4
TOD_3	[1 2 1 4 6 7 3 4 2 5 7 4 2 5 4 6 5 8 7 6 11]	1	1	2
TOD_4	[1 2 3 10 9 7 1 2 9]	3	4	3

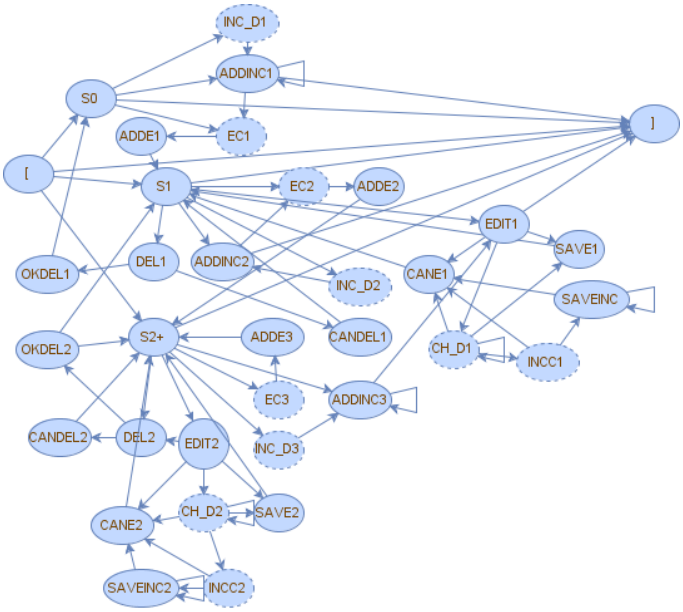
Tablo E2. 23. ODG_3 için Friedman testi sonuçları

	Test İstatistikleri
Ki-Kare	8,000
Serbestlik Derecesi	2
Anlamlılık Değeri	0,018

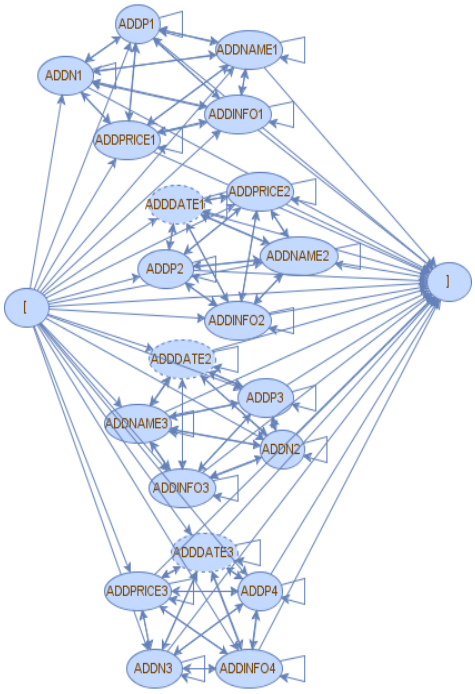
Tablo E2. 24. ODG_3 için Kendall Tau testi sonuçları

			YÖ	GO	BCO
Kendall's Tau_b	YÖ	Korelasyon katsayısı	1,000	0,913	0,667
		Anlamlılık (çift-taraflı)	0,000	0,071	0,174
	GO	Korelasyon katsayısı		1,000	0,913
		Anlamlılık (çift-taraflı)		0,000	0,071
	BCO	Korelasyon katsayısı			1,000
		Anlamlılık (çift-taraflı)			0,000

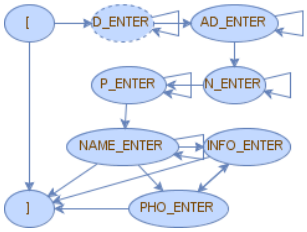
Ek 3. 1. Uygulama II'de Kullanılan Modeller



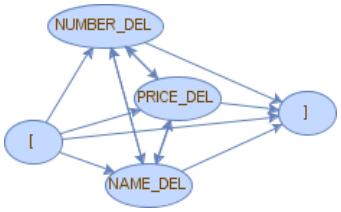
Şekil E3. 1. ODG 1: Özel indirimler modülünün ana modeli



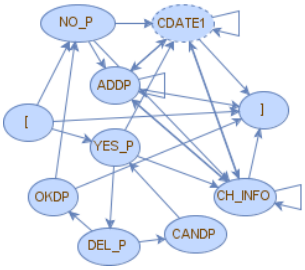
Şekil E3. 2. ODG 2 : Eksik veri girişi modeli



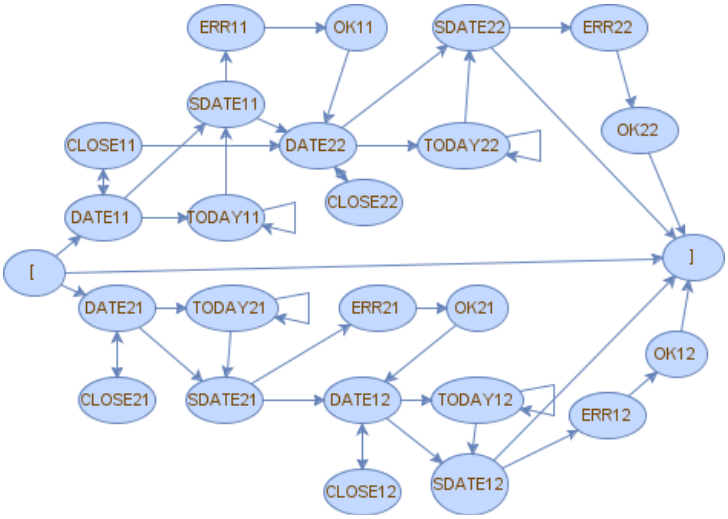
Şekil E3. 3. ODG 3 : Tam veri girişi modeli



Şekil E3. 4. ODG 4 : Veri silme modeli



Şekil E3. 5. ODG 5 : Veri değiştirme modeli



Şekil E3. 6. ODG 6: Tarih seçme modeli

Ek 3. 2. Uygulama II'de Kullanılan Test Dizileri

TOD1: 1089: [, S0, DATE21, SDATE21, DATE12, SDATE12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE1, S1, DEL1, CANDEL1, S1, DATE21, SDATE21, DATE12, SDATE12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE2, S2+, DEL2, OKDEL2, S2+, EDIT2, DEL2, CANDEL2, S2+, EDIT2, SAVE2, S2+, EDIT2, CANE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, SAVE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE22, SDATE22, CANE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE22, SDATE22, NO_P, DATE11, CLOSE11, DATE22, SDATE22, YES_P, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, SAVE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, SAVE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, CANE2, S2+, EDIT2, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, YES_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, CANE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, ERR22, OK22, CH_INFO, SAVE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, ERR22, OK22, ADDP, SAVE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, ERR22, OK22, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, ERR22, OK22, PRICE_DEL, CANE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, ERR22, OK22, NAME_DEL, CANE2, S2+, EDIT2, NO_P, DATE11, SDATE11, DATE22, SDATE22, CH_INFO, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, NO_P, DATE21, SDATE21, DATE12, SDATE12, YES_P, DATE21, SDATE21, DATE12, SDATE12, DATE11, SDATE11, DATE22, SDATE22, ADDP, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, SAVE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, YES_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, SDATE11, DATE22, SDATE22, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, CH_INFO, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, PRICE_DEL, CANE2, S2+, EDIT2, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NAME_DEL, CANE2, S2+, EDIT2, NO_P, CH_INFO, YES_P, DATE21, SDATE21, DATE12, SDATE12, CH_INFO, DATE11, SDATE11, DATE22, SDATE22, PRICE_DEL, CANE2, S2+, EDIT2, NO_P, CH_INFO, DATE21, SDATE21, DATE12, SDATE12, ADDP, YES_P, DATE21, SDATE21, DATE12, SDATE12, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, CH_INFO, CH_INFO, ADDP, DATE11, SDATE11, DATE22,

SDATE22, NAME_DEL, CANE2, S2+, EDIT2, NO_P, CH_INFO, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, ADDP, DATE21, SDATE21, DATE12, SDATE12, PRICE_DEL, CANE2, S2+, EDIT2, NO_P, ADDP, CH_INFO, PRICE_DEL, CANE2, S2+, EDIT2, NO_P, ADDP, ADDP, NUMBER_DEL, CANE2, S2+, EDIT2, NO_P, ADDP, PRICE_DEL, CANE2, S2+, EDIT2, YES_P, DATE21, SDATE21, DATE12, SDATE12, NAME_DEL, CANE2, S2+, EDIT2, YES_P, DEL_P, OKDP, SAVE2, S2+, EDIT2, YES_P, DEL_P, OKDP, CANE2, S2+, EDIT2, YES_P, DEL_P, OKDP, NO_P, ADDP, NAME_DEL, CANE2, S2+, EDIT2, YES_P, DEL_P, OKDP, YES_P, DEL_P, OKDP, NUMBER_DEL, PRICE_DEL, NUMBER_DEL, NAME_DEL, CANE2, S2+, EDIT2, YES_P, DEL_P, OKDP, PRICE_DEL, NAME_DEL, NUMBER_DEL, SAVEINC2, SAVEINC2, CANE2, S2+, EDIT2, YES_P, DEL_P, OKDP, NAME_DEL, PRICE_DEL, SAVEINC2, CANE2, S2+, EDIT2, YES_P, DEL_P, CANDP, YES_P, CH_INFO, NAME_DEL, SAVEINC2, CANE2, S2+, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ERR22, OK22, AD_ENTER, AD_ENTER, N_ENTER, N_ENTER, P_ENTER, P_ENTER, NAME_ENTER, NAME_ENTER, PHO_ENTER, INFO_ENTER, PHO_ENTER, ADDE3, S2+, DATE11, SDATE11, DATE22, SDATE22, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, INFO_ENTER, ADDE3, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE3, S2+, DATE21, SDATE21, DATE12, SDATE12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE3, S2+, DEL2, OKDEL2, S1, EDIT1, SAVE1, S1, EDIT1, CANE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, SAVE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, CANE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, NO_P, DATE11, CLOSE11, DATE22, SDATE22, YES_P, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, SAVE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, SAVE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, CANE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, YES_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, CANE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, CH_INFO, SAVE1, S1, EDIT1, NO_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP, SAVE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, NO_P, DATE21, SDATE21, DATE12, SDATE12, YES_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, NUMBER_DEL, CANE1, S1, EDIT1, NO_P, DATE21, SDATE21,

DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, PRICE_DEL, CANE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, SAVE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, CANE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, YES_P, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, NAME_DEL, CANE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, CH_INFO, CANE1, S1, EDIT1, NO_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, CH_INFO, NO_P, CH_INFO, YES_P, DATE11, CLOSE11, DATE22, SDATE22, ADDP, CANE1, S1, EDIT1, NO_P, ADDP, NO_P, ADDP, YES_P, DATE11, CLOSE11, DATE22, SDATE22, NUMBER_DEL, CANE1, S1, EDIT1, NO_P, ADDP, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, PRICE_DEL, CANE1, S1, EDIT1, NO_P, ADDP, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP, CH_INFO, DATE11, SDATE11, DATE22, SDATE22, NAME_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NUMBER_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, PRICE_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, NAME_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, CH_INFO, DATE21, SDATE21, DATE12, SDATE12, ADDP, ADDP, NUMBER_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, NUMBER_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, PRICE_DEL, CANE1, S1, EDIT1, YES_P, DATE21, SDATE21, DATE12, SDATE12, NAME_DEL, CANE1, S1, EDIT1, YES_P, DEL_P, OKDP, SAVE1, S1, EDIT1, YES_P, DEL_P, OKDP, CANE1, S1, EDIT1, YES_P, DEL_P, OKDP, NO_P, ADDP, PRICE_DEL, CANE1, S1, EDIT1, YES_P, DEL_P, OKDP, YES_P, DEL_P, OKDP, NUMBER_DEL, CANE1, S1, EDIT1, YES_P, DEL_P, OKDP, PRICE_DEL, CANE1, S1, EDIT1, YES_P, DEL_P, OKDP, NAME_DEL, CANE1, S1, EDIT1, YES_P, DEL_P, CANDP, YES_P, CH_INFO, CH_INFO, ADDP, NAME_DEL, CANE1, S1, EDIT1, YES_P, CH_INFO, NUMBER_DEL, PRICE_DEL, NUMBER_DEL, NAME_DEL, NUMBER_DEL, SAVEINC, SAVEINC, CANE1, S1, EDIT1, YES_P, CH_INFO, PRICE_DEL, NAME_DEL, PRICE_DEL, SAVEINC, CANE1, S1, EDIT1, YES_P, CH_INFO, NAME_DEL, SAVEINC, CANE1, S1, DEL1, OKDEL1, S0,],

TOD2 2: [, S0, ADDINC1,],

TOD3 28: [, S0, ADDN1, ADDN1, ADDPRICE1, ADDPRICE1, ADDP1, ADDP1, ADDNAME1, ADDNAME1, ADDINFO1, ADDINFO1, ADDNAME1, ADDP1, ADDINFO1, ADDP1, ADDPRICE1, ADDNAME1, ADDPRICE1, ADDINFO1, ADDPRICE1, ADDN1, ADDP1, ADDN1, ADDNAME1, ADDN1, ADDINFO1, ADDN1, ADDINC1,],

TOD4 3: [, S0, ADDPRICE1, ADDINC1,],

TOD5 3: [, S0, ADDP1, ADDINC1,],

TOD6 3: [, S0, ADDNAME1, ADDINC1,],

TOD7 3: [, S0, ADDINFO1, ADDINC1,],

TOD8 146: [, S0, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP2, ADDPRICE2, ADDPRICE2, ADDNAME2, ADDPRICE2, ADDP2, ADDNAME2, ADDNAME2, ADDP2, ADDP2, ADDINFO2, ADDINFO2, ADDPRICE2, ADDINFO2, ADDNAME2, ADDINFO2, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE2, DATE21, SDATE21, DATE12, SDATE12, ADDNAME2, DATE21, SDATE21, DATE12, SDATE12, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO2, DATE11, CLOSE11, DATE22, SDATE22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME2, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDP2, DATE11, SDATE11, DATE22, SDATE22, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME2, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC1,],

TOD9 16: [, S0, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC1,],

TOD10 24: [, S0, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC1,],

TOD11 14: [, S0, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC1,],

TOD12 3: [, S0, ADDPRICE2, ADDINC1,],

TOD13 3: [, S0, ADDNAME2, ADDINC1,],

TOD14 3: [, S0, ADDP2, ADDINC1,],

TOD15 3: [, S0, ADDINFO2, ADDINC1,],

TOD16 146: [, S0, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN2, ADDNAME3, ADDNAME3, ADDP3, ADDNAME3, ADDN2, ADDP3, ADDP3, ADDN2, ADDN2, ADDINFO3, ADDINFO3, ADDNAME3, ADDINFO3, ADDP3, ADDINFO3, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ADDNAME3, DATE21, SDATE21, DATE12, SDATE12, ADDP3, DATE21, SDATE21, DATE12, SDATE12, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP3,

DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO3, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ADDP3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDN2, DATE11, SDATE11, DATE22, SDATE22, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP3, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC1,],

TOD17 16: [, S0, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC1,],

TOD18 24: [, S0, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC1,],

TOD19 14: [, S0, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC1,],

TOD20 3: [, S0, ADDNAME3, ADDINC1,],

TOD21 3: [, S0, ADDP3, ADDINC1,],

TOD22 3: [, S0, ADDN2, ADDINC1,],

TOD23 3: [, S0, ADDINFO3, ADDINC1,],

TOD24 146: [, S0, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE3, ADDP4, ADDP4, ADDN3, ADDP4, ADDPRICE3, ADDN3, ADDN3, ADDPRICE3, ADDPRICE3, ADDINFO4, ADDINFO4, ADDP4, ADDINFO4, ADDN3, ADDINFO4, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ADDP4, DATE21, SDATE21, DATE12, SDATE12, ADDN3, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDPRICE3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO4, DATE11, CLOSE11, DATE22, SDATE22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ADDN3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDPRICE3, DATE11, SDATE11, DATE22, SDATE22, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN3, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC1,],

TOD25 16: [, S0, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC1,],

TOD26 24: [, S0, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC1,],

TOD27 14: [, S0, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC1,],

TOD28 3: [, S0, ADDP4, ADDINC1,],

TOD29 3: [, S0, ADDN3, ADDINC1,],

TOD30 77: [, S0, ADDPRICE3, ADDINC1, ADDINC1, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ERR22, OK22, AD_ENTER, AD_ENTER, N_ENTER, N_ENTER, P_ENTER, P_ENTER, NAME_ENTER, NAME_ENTER, PHO_ENTER, INFO_ENTER, PHO_ENTER, ADDE1, S1,],

TOD31 23: [, S0, ADDINFO4, ADDINC1, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, INFO_ENTER, ADDE1, S1, ADDINC2,],

TOD32 38: [, S0, DATE11, SDATE11, DATE22, SDATE22, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE1, S1, ADDN1, ADDN1, ADDPRICE1, ADDPRICE1, ADDP1, ADDP1, ADDNAME1, ADDNAME1, ADDINFO1, ADDINFO1, ADDNAME1, ADDP1, ADDINFO1, ADDP1, ADDPRICE1, ADDNAME1, ADDPRICE1, ADDINFO1, ADDPRICE1, ADDN1, ADDP1, ADDN1, ADDNAME1, ADDN1, ADDINFO1, ADDN1, ADDINC2,],

TOD33 3: [, S1, ADDPRICE1, ADDINC2,],

TOD34 3: [, S1, ADDP1, ADDINC2,],

TOD35 3: [, S1, ADDNAME1, ADDINC2,],

TOD36 3: [, S1, ADDINFO1, ADDINC2,],

TOD37 146: [, S1, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP2, ADDPRICE2, ADDPRICE2, ADDNAME2, ADDPRICE2, ADDP2, ADDNAME2, ADDNAME2, ADDP2, ADDP2, ADDINFO2, ADDINFO2, ADDPRICE2, ADDINFO2, ADDNAME2, ADDINFO2, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE2, DATE21, SDATE21, DATE12, SDATE12, ADDNAME2, DATE21, SDATE21, DATE12, SDATE12, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE2, DATE11, CLOSE11, DATE22,

SDATE22, ERR22, OK22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO2, DATE11, CLOSE11, DATE22, SDATE22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME2, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDP2, DATE11, SDATE11, DATE22, SDATE22, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME2, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC2,],

TOD38 16: [, S1, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC2,],

TOD39 24: [, S1, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC2,],

TOD40 14: [, S1, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC2,],

TOD41 3: [, S1, ADDPRICE2, ADDINC2,],

TOD42 3: [, S1, ADDNAME2, ADDINC2,],

TOD43 3: [, S1, ADDP2, ADDINC2,],

TOD44 3: [, S1, ADDINFO2, ADDINC2,],

TOD45 146: [, S1, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN2, ADDNAME3, ADDNAME3, ADDP3, ADDNAME3, ADDN2, ADDP3, ADDP3, ADDN2, ADDN2, ADDINFO3, ADDINFO3, ADDNAME3, ADDINFO3, ADDP3, ADDINFO3, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ADDNAME3, DATE21, SDATE21, DATE12, SDATE12, ADDP3, DATE21, SDATE21, DATE12, SDATE12, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO3, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ADDP3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDN2, DATE11, SDATE11, DATE22, SDATE22, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP3, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC2,],

TOD46 16: [, S1, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC2,],

TOD47 24: [, S1, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC2,],

TOD48 14: [, S1, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC2,],

TOD49 3: [, S1, ADDNAME3, ADDINC2,],

TOD50 3: [, S1, ADDP3, ADDINC2,],

TOD51 3: [, S1, ADDN2, ADDINC2,],

TOD52 3: [, S1, ADDINFO3, ADDINC2,],

TOD53 146: [, S1, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE3, ADDP4, ADDP4, ADDN3, ADDP4, ADDPRICE3, ADDN3, ADDN3, ADDPRICE3, ADDPRICE3, ADDINFO4, ADDINFO4, ADDP4, ADDINFO4, ADDN3, ADDINFO4, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ADDP4, DATE21, SDATE21, DATE12, SDATE12, ADDN3, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDPRICE3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO4, DATE11, CLOSE11, DATE22, SDATE22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ADDN3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDPRICE3, DATE11, SDATE11, DATE22, SDATE22, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN3, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, ADDINC2,],

TOD54 16: [, S1, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, ADDINC2,],

TOD55 24: [, S1, DATE21, SDATE21, DATE12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC2,],

TOD56 14: [, S1, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC2,],

TOD57 3: [, S1, ADDP4, ADDINC2,],

TOD58 3: [, S1, ADDN3, ADDINC2,],

TOD59 76: [, S1, ADDPRICE3, ADDINC2, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ERR22, OK22, AD_ENTER, AD_ENTER, N_ENTER, N_ENTER, P_ENTER, P_ENTER, NAME_ENTER, NAME_ENTER, PHO_ENTER, INFO_ENTER, PHO_ENTER, ADDE2, S2+,],

TOD60 34: [, S1, ADDINFO4, ADDINC2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, INFO_ENTER, ADDE2, S2+, ADDN1, ADDN1, ADDPRICE1, ADDN1, ADDP1, ADDN1, ADDNAME1, ADDN1, ADDINFO1, ADDN1, ADDINC3, ADDINC3,],

TOD61 20: [, S1, DATE11, SDATE11, DATE22, SDATE22, AD_ENTER, N_ENTER, P_ENTER, NAME_ENTER, ADDE2, S2+, ADDPRICE1, ADDPRICE1, ADDP1, ADDPRICE1, ADDNAME1, ADDPRICE1, ADDINFO1, ADDPRICE1, ADDINC3,],

TOD62 8: [, S2+, ADDP1, ADDP1, ADDNAME1, ADDP1, ADDINFO1, ADDP1, ADDINC3,],

TOD63 6: [, S2+, ADDNAME1, ADDNAME1, ADDINFO1, ADDNAME1, ADDINC3,],

TOD64 4: [, S2+, ADDINFO1, ADDINFO1, ADDINC3,],

TOD65 86: [, S2+, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC3,],

TOD66 50: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ADDPRICE2, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME2, DATE11, CLOSE11, DATE22, SDATE22, ADDP2, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME2, DATE11, SDATE11, DATE22, SDATE22, ADDINC3,],

TOD67 22: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC3,],

TOD68 26: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE2, DATE21, SDATE21, DATE12, SDATE12, ADDNAME2, DATE21, SDATE21, DATE12, SDATE12, ADDP2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO2, DATE21, SDATE21, DATE12, SDATE12, ADDINC3,],

TOD69 10: [, S2+, ADDPRICE2, ADDPRICE2, ADDNAME2, ADDPRICE2, ADDP2, ADDPRICE2, ADDINFO2, ADDPRICE2, ADDINC3,],

TOD70 8: [, S2+, ADDNAME2, ADDNAME2, ADDP2, ADDNAME2, ADDINFO2, ADDNAME2, ADDINC3,],

TOD71 6: [, S2+, ADDP2, ADDP2, ADDINFO2, ADDP2, ADDINC3,],

TOD72 4: [, S2+, ADDINFO2, ADDINFO2, ADDINC3,],

TOD73 86: [, S2+, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN2, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC3,],

TOD74 50: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDNAME3, DATE11, CLOSE11, DATE22, SDATE22, ADDNAME3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDP3, DATE11, SDATE11, DATE22, SDATE22, ADDN2, DATE11, SDATE11, DATE22, SDATE22, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP3, DATE11, SDATE11, DATE22, SDATE22, ADDINC3,],

TOD75 22: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC3,],

TOD76 26: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ADDNAME3, DATE21, SDATE21, DATE12, SDATE12, ADDP3, DATE21, SDATE21, DATE12, SDATE12, ADDN2, DATE21, SDATE21, DATE12, SDATE12, ADDINFO3, DATE21, SDATE21, DATE12, SDATE12, ADDINC3,],

TOD77 10: [, S2+, ADDNAME3, ADDNAME3, ADDP3, ADDNAME3, ADDN2, ADDNAME3, ADDINFO3, ADDNAME3, ADDINC3,],

TOD78 8: [, S2+, ADDP3, ADDP3, ADDN2, ADDP3, ADDINFO3, ADDP3, ADDINC3,],

TOD79 6: [, S2+, ADDN2, ADDN2, ADDINFO3, ADDN2, ADDINC3,],

TOD80 4: [, S2+, ADDINFO3, ADDINFO3, ADDINC3,],

TOD81 86: [, S2+, DATE11, CLOSE11, DATE11, CLOSE11, DATE22, CLOSE22, DATE22, TODAY22, TODAY22, SDATE22, DATE11, CLOSE11, DATE22, SDATE22, DATE21, CLOSE21, DATE21, TODAY21, TODAY21, SDATE21, ERR21, OK21, DATE12, CLOSE12, DATE12, TODAY12, TODAY12, SDATE12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, DATE21, SDATE21, DATE12, SDATE12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDN3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDPRICE3, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINFO4, DATE11, CLOSE11, DATE22, SDATE22, ERR22, OK22, ADDINC3,],

TOD82 50: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ADDP4, DATE11, CLOSE11, DATE22, SDATE22, ADDN3, DATE11, CLOSE11, DATE22, SDATE22, ADDPRICE3, DATE11, TODAY11, TODAY11, SDATE11, ERR11, OK11, DATE22, SDATE22, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDN3, DATE11, SDATE11, DATE22, SDATE22, ADDINC3,],

TOD83 22: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ERR12, OK12, ADDINC3,],

TOD84 26: [, S2+, DATE21, SDATE21, DATE12, SDATE12, ADDP4, DATE21, SDATE21, DATE12, SDATE12, ADDN3, DATE21, SDATE21, DATE12, SDATE12, ADDPRICE3, DATE21, SDATE21, DATE12, SDATE12, ADDINFO4, DATE21, SDATE21, DATE12, SDATE12, ADDINC3,],

TOD85 10: [, S2+, ADDP4, ADDP4, ADDN3, ADDP4, ADDPRICE3, ADDP4, ADDINFO4, ADDP4, ADDINC3,],

TOD86 8: [, S2+, ADDN3, ADDN3, ADDPRICE3, ADDN3, ADDINFO4, ADDN3, ADDINC3,],

TOD87 6: [, S2+, ADDPRICE3, ADDPRICE3, ADDINFO4, ADDPRICE3, ADDINC3,],

TOD88 4: [, S2+, ADDINFO4, ADDINFO4, ADDINC3,],

TOD89 2: [, S2+, ADDINC3,],

Ek 3. 3. Yazılıma Eklenen Hatalar

Tablo E3. 1. Yazılıma eklenen ve test sırasında yakalanan hatalar

Hata	Açıklama
Product_stepFiveOperations Line 47 (satır silindi)	Verileri eksiksiz girsen bile special eklemiyor (Delete Return) ve girdiğin specialın bilgileri ekranda kalıyor
Specials Line 126 (satır silindi)	Specialı eklemiyor. "Data could not be saved" hatasını veriyor . (Delete Return)
Specials Line 136 (satır silindi)	Specialı silmiyor "error while deleting" hatası veriyor(Delete Return)
Specials Line 214 (satır silindi)	Foto eklemiyor. (Delete Return)
Product_stepFiveOperations Line 230 (satır silindi)	ADD butonuna bir kere basınca bir daha aktif olmuyor (Delete assigment or statement)
Product_stepFiveOperations Line 259 (satır silindi)	Editten sonra save butonuna basınca specialnameexists functionunda hata veriyor. Fatal error (Delete assigment or statement)
Product_stepFiveOperations Line 274 (satır silindi)	Editten sonra save tuşuna basında var olan tüm specialları siliyor ve add butonu aktif olmuyor (Delete Statement)
Product_stepFiveOperations Line 299 (satır silindi)	Delete tuşuna basınca tüm specialları siliyor ve add butonu aktif olmuyor(Delete Statement)
Product_stepFiveOperations Line 322 (satır silindi)	Edit tuşuna basınca tüm specialları siliyor ve add butonu aktif olmuyor(Delete Statement)
Product_stepFiveOperations Line 318 (satır silindi)	Edit tuşuna basınca edit ekranı gelmiyor. Ekran aynı kalıyor. (Delete assigment or statement)
Product_stepFiveOperations Line 339 (satır silindi)	Editten sonra cancel tuşuna basınca tüm specialları siliyor ve add aktif olmuyor(Delete Statement)
Product_stepFiveOperations Line 287 (satır silindi)	Specialı silmiyor "error while deleting" hatası veriyor(Delete assigment or statement)
Product_stepFiveOperations Line 268 (satır silindi)	Editten sonra save tuşuna basınca add special menüsünde bazı bilgiler dolu halde geliyor(Delete Statement)
Product_stepFiveOperations Line 354 (satır silindi)	Foto silince tüm speaciallar siliniyor. (Delete Statement)
Specials Line 44(satır silindi)	Eksiksiz veri girsen bile specialı eklemiyor "This functionality is not fully programmed" ve "Data could not be saved" hatalarını veriyor(Delete Statement)
Specials Line 45 (satır silindi)	Eksiksiz veri girsen bile specialı eklemiyor "This functionality is not fully programmed" ve "Data could not be saved" hatalarını veriyor(Delete Statement)
Specials Line 88 (satır silindi)	Eksiksiz veri girince specialı ekliyor fakat delete ve edit butonları yerine "in process" yazısı geliyor. (Delete assigment or statement)
Specials Line 91 (satır silindi)	Add specials to list ekranında resim eklemiyor ama editte ekliyor. (Delete Statement) ve resim eklemesen de eklenen specialın yanında resim simgesi görülüyor
Specials Line 100 (satır silindi)	Specialı foto ile birlikte eklersen yada edit menüsünde foto eklersen hata veriyor ve eklemiyor. This functionality is not fully programmed" ve "Data could not be saved" (Delete assigment or statement)
Product_stepFiveOperations Line 317 (satır silindi)	Edit tuşuna basınca edit ekranı gelmiyor. Ekran aynı kalıyor. " (Delete assigment or statement)
Product_stepFiveOperations Line 34 (satır silindi)	Başlangıç tarihini girmesen de specialı ekliyor. " (Delete assigment or statement)
Product_stepFiveOperations Line 247 (satır silindi)	Add tuşuna basınca tüm specialları siliyor ve ADD tuşu pasif oluyor (Delete Statement)

Tablo E3. 1. **(Devamı…)**

Hata	Açıklama
SpecialsController line149 (satır silindi)	Edit bölümünde değişiklik yaptığım halde değişikliği kabul etmiyor()(Delete assigment or statement)
Product_stepFiveOperations Line 17	Eksiksiz veri girsen bile specialı eklemiyor. True--False
Product_stepFiveOperations Line 252	Eksik veri ile special ekliyor True--False
Product_stepFiveOperations Line 279	Edit butonundan sonra eksik veri ile bile save ediyor True--False
Product_stepFiveOperations Line 328	Edit butonundan sonra eksik veri ile bile save ediyor True--False
Product_stepFiveOperations Line 23	Aynı isme sahip speciallar girmene izin veriyor. Ancak yine de "name already exits" hatasını veriyor Editten sonra save yapıncada aynı hatayı veriyor False--True
Product_stepFiveOperations Line 38	Tarih aralıklarını girmeden special girmene izin veriyor ve aynı isime sahip speciallar girilebiliyor(tarih aralıkları girilmediğinde) False--True
Product_stepFiveOperations Line 42	Birden fazla special girmene izin vermiyor. Maximum specials of 10 allowed hatası veriyor Integer Replacement(10-→1)
Specials Line 328	Integer Replacement(100→1)
Product_stepFiveOperations Line 42	Hiç special girmene izin vermiyor "A maximum of 10 specials I allowed" hatasını veriyor &&--‖
Product_stepFiveOperations Line 32	Bitiş tarihini girmeden de speacial eklemene izin veriyor. >=--<=
Product_stepFiveOperations Line 42	Hiç special girmene izin vermiyor "A maximum of 10 specials I allowed" hatasını veriyor >=--<=
SpecialsController Line 106	Sen hangisi specialın Edit butonuna basarsan bas ilk specialın edit ekranı geliyor = = →!=
SpecialsController Line 162	Aynı isime sahip speciallar girmene izin veriyor != → = =
SpecialsController Line 133	Silmek için hangi specialı seçersen seç ilk specialı siliyor == → !=
Specials Line 146	Foto delete tuşuna bastığında cancel tuşuna basınca fotoyu siliyor. == → !=
Product_stepFiveOperations Line 234	Tüm verileri eksiksiz girsen bile specialı eklemiyor (X&&Y)→(!X&&Y)
Product_stepFiveOperations Line 261	Save tuşuna bastığında "Add to special to list" ekranı gelmiyor edit ekranında kalıyor (If →If!
SpecialsController Line 106	Fatal Error: Call to a member function asArray() …If →If!
SpecialsController Line 117	Verileri eksiksiz girsende specialı eklemiyor ve "Data could nor be saved " hatası veriyor If →If!
SpecialsController Line 133	Specialı silmiyor "Error while deleting" hatası veriyor If →If!
SpecialsController Line 150	Editten sonra save tuşuna basınca "Data could not be saved" hatası veriyor If →If!
Yakalanan hata	Today butonuna tıklandığında bugün ki tarih seçilmiyor
Yakalanan hata	Geçmiş tarihli kayıt yapılmasına izin veriliyor
Yakalanan hata	Geçmiş tarihli kayıt girilmek istendiğinde hata mesajı görünmüyor
Yakalanan hata	Eksik hata mesajı
Yakalanan hata	Çakışan tarihlerde ard arda iki hata veriyor.

Ek 3. 4. Kendall Tau-b İstatistiklerinin Hesaplanmasında Kullanılan Veri Tabloları

Tablo E3. 2. HYOK kriterine göre ideal durum ile test önceliklendirme metotlarıyla elde edilen sonuçların karşılaştırılması

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
1	22	22	22	22	22
2	14	10	10	10	10
3	14	11	10	10	10
4	11	10	10	11	10
5	11	10	10	10	10
6	10	10	11	10	11
7	10	10	10	10	10
8	10	9	9	9	9
9	10	11	11	11	10
10	10	10	10	10	11
11	10	14	14	14	14
12	9	14	14	14	14
13	9	9	8	8	9
14	8	8	9	9	8
15	7	7	7	7	7
16	6	5	6	6	4
17	6	5	4	5	6
18	6	4	6	5	1
19	6	5	3	4	3
20	6	5	3	5	3
21	6	5	3	5	3
22	6	3	5	5	5
23	6	3	5	3	5
24	5	3	5	3	5
25	5	4	5	3	5
26	5	4	5	4	5
27	5	4	4	4	6
28	5	6	4	4	4
29	5	6	4	4	4
30	5	6	4	6	4
31	5	6	6	6	4
32	5	6	6	6	6
33	5	6	4	6	6
34	4	6	6	6	4
35	4	4	6	6	6
36	4	6	6	4	6
37	4	5	4	6	6
38	4	5	5	5	5
39	4	5	5	5	5
40	4	5	6	5	5
41	3	4	5	5	6
42	3	4	5	4	5
43	3	1	1	1	4
44	3	1	1	1	5
45	1	1	1	1	1
46	1	1	1	1	1
47	1	1	1	1	1
48	1	1	1	1	1

Tablo E3. 2. (Devamı…)

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
49	1	1	1	1	1
50	1	1	1	1	1
51	1	1	1	1	1
52	1	1	1	1	1
53	1	1	1	1	1
54	1	1	1	1	1
55	1	1	1	1	1
56	1	1	1	1	1
57	1	1	1	1	1
58	1	1	1	1	1
59	1	1	1	1	1
60	1	1	1	1	1
61	1	1	1	1	1
62	1	1	1	1	6
63	1	1	1	1	1
64	1	1	1	1	1
65	1	1	3	1	1
66	1	1	1	1	1
67	1	1	1	1	1
68	1	1	1	1	1
69	1	1	1	1	1
70	1	1	1	1	1
71	1	1	1	1	3
72	1	1	1	1	1
73	1	1	1	1	1
74	1	3	1	1	1
75	1	1	1	1	1
76	1	1	1	1	1
77	1	1	1	1	1
78	1	1	1	1	1
79	1	1	1	3	1
80	1	1	1	1	1
81	1	1	1	1	1
82	1	1	1	1	1
83	1	1	1	1	1
84	1	1	1	1	1
85	1	1	1	1	1
86	1	1	1	1	1
87	1	1	1	1	1
88	1	1	5	5	1
89	1	5	1	1	1

Tablo E3. 3. TRHY kriterine göre ideal durum ile test önceliklendirme metotlarıyla elde edilen sonuçların karşılaştırılması

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
1	54	54	54	54	54
2	32	12	12	12	15
3	20	15	12	12	15
4	18	15	12	12	15
5	15	12	15	15	12
6	15	12	15	15	12
7	15	15	15	15	12
8	13	10	10	10	10
9	12	13	13	13	11
10	12	11	11	11	13
11	12	18	18	18	32
12	11	32	32	32	18
13	10	10	10	9	10
14	10	9	9	10	9
15	9	8	8	8	8
16	9	2	5	7	2
17	9	2	5	5	7
18	9	2	4	5	3
19	9	7	8	4	2
20	9	8	8	8	2
21	8	8	8	8	2
22	8	8	2	8	8
23	8	5	2	2	8
24	8	5	2	2	8
25	7	4	4	2	5
26	7	2	4	4	5
27	7	20	4	4	20
28	6	4	7	4	4
29	6	4	2	2	4
30	6	4	20	20	4
31	5	5	6	6	5
32	5	9	5	5	9
33	5	9	5	5	9
34	5	7	7	9	7
35	5	6	6	9	6
36	5	5	6	7	5
37	4	5	9	7	5
38	4	9	9	6	9
39	4	9	7	6	9
40	4	7	9	9	9
41	3	6	9	9	7
42	3	6	3	5	6
43	3	3	5	3	4
44	3	0	0	0	6
45	3	0	0	0	0
46	3	0	0	0	0
47	3	0	0	0	0
48	3	0	0	0	0
49	3	0	0	0	0
50	3	0	0	0	0

Tablo E3. 3. (Devamı…)

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
51	3	0	0	0	0
52	3	0	0	0	0
53	3	0	0	0	0
54	3	0	0	0	0
55	3	0	0	0	0
56	2	0	0	0	0
57	2	0	0	0	0
58	2	0	0	0	0
59	2	3	0	0	0
60	0	3	0	0	3
61	0	3	0	0	3
62	0	3	0	0	3
63	0	3	0	0	3
64	0	3	0	0	3
65	0	3	0	0	3
66	0	3	0	0	3
67	0	3	0	0	3
68	0	3	0	0	3
69	0	3	0	0	3
70	0	5	0	0	3
71	0	3	0	0	5
72	0	3	3	0	3
73	0	0	0	3	3
74	0	0	3	3	0
75	0	0	3	3	0
76	0	0	3	3	0
77	0	0	3	3	0
78	0	0	3	3	0
79	0	0	3	5	0
80	0	0	3	3	0
81	0	0	3	3	0
82	0	0	3	3	0
83	0	0	3	3	0
84	0	0	5	3	0
85	0	0	3	3	0
86	0	0	3	3	0
87	0	3	0	3	0
88	0	9	3	9	3
89	0	0	9	0	0

Tablo E3. 4. TZHY kriterine göre ideal durum ile test önceliklendirme metotlarıyla elde edilen sonuçların karşılaştırılması

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
1	47	47	47	47	47
2	16	7	7	7	7
3	16	7	7	7	7
4	16	7	7	7	7
5	10	7	7	7	7
6	10	7	7	7	7
7	10	7	7	7	7
8	7	6	6	6	6
9	7	10	10	10	10
10	7	10	10	10	10
11	7	10	10	10	16
12	7	16	16	16	10
13	7	7	3	3	7
14	7	3	7	7	3
15	6	3	3	3	3
16	6	2	6	6	16
17	6	2	16	2	6
18	5	2	16	2	1
19	5	2	3	2	3
20	5	2	3	2	3
21	5	2	3	2	3
22	5	3	2	2	2
23	5	3	2	3	2
24	3	3	2	3	2
25	3	3	2	3	2
26	3	3	2	3	2
27	3	3	2	3	16
28	3	6	3	3	3
29	3	16	3	16	3
30	3	5	3	16	3
31	3	5	5	5	6
32	2	5	5	5	5
33	2	5	5	5	5
34	2	5	5	5	5
35	2	5	5	5	5
36	2	2	5	5	5
37	2	2	6	2	5
38	2	2	2	2	2
39	2	2	2	2	2
40	2	2	2	2	2
41	2	16	2	2	2
42	2	6	2	6	2
43	2	1	1	1	2
44	1	1	1	1	2
45	1	1	1	1	1
46	1	1	1	1	1
47	1	0	1	1	1
48	1	1	0	0	1
49	1	1	1	1	1
50	1	1	1	1	0

Tablo E3. 4. (Devamı…)

Öncelik Sırası	İdeal	GO	YO	BCO	Uzunluk
51	1	0	0	1	1
52	1	1	1	0	1
53	1	1	1	1	1
54	1	1	1	1	0
55	1	1	1	1	1
56	1	0	0	0	1
57	0	0	1	1	1
58	0	0	0	1	0
59	0	0	0	0	1
60	0	0	0	0	0
61	0	0	0	0	0
62	0	1	0	0	0
63	0	0	0	0	0
64	0	0	0	0	0
65	0	0	0	0	0
66	0	0	0	0	0
67	0	0	0	0	0
68	0	0	0	0	0
69	0	0	0	0	0
70	0	1	0	0	0
71	0	0	0	0	0
72	0	0	0	0	0
73	0	0	0	0	0
74	0	0	0	0	0
75	0	0	0	0	0
76	0	0	0	0	0
77	0	0	0	0	0
78	0	0	1	0	0
79	0	0	0	0	0
80	0	0	0	0	0
81	0	0	0	0	0
82	0	0	0	0	0
83	0	0	0	0	0
84	0	0	0	0	0
85	0	0	0	0	0
86	0	0	0	0	0
87	0	0	0	0	0
88	0	0	2	2	0
89	0	2	0	0	0

ÖZGEÇMİŞ

Kişisel Bilgiler	Adı Soyadı	: Nida GÖKÇE
	Medeni durum	: Bekar
	Doğum Tarihi	: 21.11.1976
	Doğum Yeri	: Polatlı /ANKARA
Eğitim	Lise	: Polatlı Lisesi / 1990-1993
	Lisans	: Muğla Üniversitesi, Fen-Edebiyat Fakültesi, İstatistik ve Bilgisayar Bilimler, 1996-2001
	Y.Lisans	: Muğla Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik ve Bilgisayar Bilimleri Bölümü 2002-2005
	Doktora	: Muğla Üniversitesi, Fen Bilimleri Enstitüsü, Matematik A.B.D. 2006-2011
Yabancı Diller	İngilizce, Almanca	
İş Deneyimi	Muğla Üniversitesi, Fen Bilimleri Enstitüsü, Araştırma Görevlisi, 2002-2003	
	Muğla Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Araştırma Görevlisi, 2003- devam ediyor	

Yayınlar

- Eminov, M., Gökçe, N., 2005. *Neural Network Clustering Using Competitive Learning Algorithm*, Proc, TAINN 2005, 161-168p.
- Gökçe, N., Eminov, M., Belli, F., 2006. *Coverage-Based, Prioritized Testing Using Neural Network Clustering*, The 21st International Symposium on Computer and Information Sciences, ISCIS 2006 Proceedings, LNCS Vol.4263. 1060-1071p.
- Belli, F., Eminov, M., Gökçe, N., 2007a. *Prioritizing Coverage-Oriented Testing Process- An Adaptive-Learning-Based Approach and Case Study*, 31st Annual International Computer Software and Applications Conference, COMPSAC 2007, Vol. 2. 197-203p.
- Belli, F., Eminov, M., Gökçe, N., 2007b, *Coverage-Oriented, Prioritized Testing – A Fuzzy Clustering Approach and Case Study*, Springer Berlin, LNCS Volume 4746, 95-110p.

- Belli, F., Eminov, M., Gökçe, N., 2009. *Model-Based Test Prioritization- a Comparative Soft- Computing Approach and Case Studies*, LNCS, Vol. 5803, 427-434p.
- Belli, F. Gökçe, N., 2010. *Test Prioritization at Different Modeling Levels*, International Conference, ASEA 2010, Communications in Computer and Information Science, Vol. 117, 130-140p.
- Belli, F., Eminov, M., Gökçe, N., Wong, E., 2011. *Prioritizing Coverage-Oriented Testing Process — An Adaptive-Learning-Based Approach and Case Study*, Series on Software Engineering and Knowledge Engineering, Vol. 20, Adaptive Control Approach For Software Quality Improvement, 1-21p.